1 Introduction

Consumption of modern-day technology is ever increasing. As a result, the risk of an individual, or business, becoming a victim to cyber-crime increases proportionally. Small and medium-sized businesses (SMBs) are the biggest sectors targeted by cyber-criminals [1]. This stems from issues such as budget restraints and a general lack of understanding regarding cyber security concepts [1, 2].

In fact, because COVID-19 changed the dynamic of industry standards this past year, a report from May 2020 (UK) [3] showed that individuals experiencing targeted hacking increased by 77.41% - in comparison with the previous year. As employees are encouraged to work from home, this has fed into a new strategy whereby cyber-criminals are moving laterally into organisational infrastructure by targeting employees at their less secure personal computers [4].

Regarding this problem, this paper will explore the effectiveness of educational games as these have been shown to have improved learning outcomes in comparison with traditional training material [5].

Problem Statement

Despite the existence of many cyber security awareness programs, there is a lack of widespread cyber security awareness

Research Question

Can teaching cyber security through a gamified medium improve cyber security awareness?

Hypothesis

The MDA framework will satisfy the educational requirements for creating a serious game

1.1 Goals

The goal of this project is to build an online multiplayer game which represents the OWASP Top Ten [6] security risks and can be used for both recreational and training purposes.

Application specific goals:

- The application will support 2-person multiplayer
- The application will be accessible, and playable, cross-platform
- The application will illustrate a range of cyber security attacks and mitigations
- The application will be extensible to support new scenarios and mechanics

1.2 Scope

As cyber-security is an incredibly broad field, only the OWASP Top Ten [6] will be used to create an educational scenario of defending and attacking web applications. Unity will be used to develop the application, in conjunction with Photon Cloud to host multiplayer scenarios. This project will be designed with the MDA framework [7] in mind.

2 Literature Review

This section presents the current problem with cyber security training methodologies. From this, gamification mechanics, cyber security games and related work are reviewed to identify a potential solution.

2.1 The Problem with Cyber Security Training Programs

Regarding the delivery of training programs, generalised security advice has little effect on changing the behaviour of employees within SMBs as they fail to relate the information to their workplace-environment [1, 2]. Furthermore, non-scenario-specific training fails to demonstrate how security risks link together in a multifaceted social engineering attack [2]. Lastly, training programs are generally undertaken in a formal setting which fosters a situation of recipients not absorbing the information effectively; this is because they aim to finish the material as fast as possible [2].

To summarise, the 4 core problems for SMBs [1, 8] are:

1. SMBs can be heavily constrained by a limited budget
2. SMBs are difficult to reach because they do not understand the severity of data breaches
3. SMBs are often distracted by the operational requirements for setting up and running a small business
4. SMBs struggle to identify their assets in terms of the risks associated with them

From this, it is evident that a light-weight training application that can be tailored for multiple scenarios would be an appropriate solution for training these employees. Furthermore, delivering the training material through a gamified medium should lead to an increased rate of information absorption - even in a workplace environment.

2.2 Why Use a Game-Based Learning Approach?

Many of the challenges from traditional training programs can be overcome with game-based learning strategies [5]. A summary of motivations for these strategies [5, 9, 10, 11, 12] include:

- Rapid progress paired with instant feedback
- Strong user engagement
- Allows the user to learn from mistakes in a safe, non-punishing environment which would otherwise discourage exploration (due to the fear of failure)
- Encourages self-learning at the user's desired pace
- Deeply immerses the player into the game world whereby learning feels like a secondary objective
- Low cost to produce in comparison to larger training schemes
- Requires little to no supervision
- Easy to distribute across multiple platforms and incorporate into the workplace
- Easy to integrate within events such as hackathons and other cyber security awareness gatherings
- An integrated rewards system such as badges and hidden achievements lead to a desire to win
- Contextually appropriate by matching the real world

2.3 Appropriate Gamification Mechanics

To evaluate pre-existing cyber security games in the next section, Forde's report [13] identifies the following gamification mechanics which increases the adoption rate of cyber security trends within the workplace:

- Avatar & User Profile
- Badges & Privileges
- Challenge
- Competition
- Collaboration
- Feedback & Guidance
- Goals & Objectives
- Incentives & Rewards
- Leader boards
- Points System
- Progress & Levels
- Role Playing
- Story
- Tips & Hints System

2.4 A Review of Pre-Existing Cyber Security Games

Table 1 presents an analysis of educational cyber security games.

Game	Game Type	Description & Key Findings	Target Audience	Mechanics
Game of Threats [14]	Multiplatform (Mobile, PC)	Employees are split into teams of attackers and defenders to simulate scenarios of cyber-attacks and appropriate responses. This prompts discussion which popularises cyber security awareness.	Business Employees	Feedback, Incentives, Competition
Cybersecurity Lab [15]	Web-based Point-and-Click	Comprised of 3 mini games which teach children how to spot phishing emails, construct strong passwords and simple programming principles.	Children (aged 7-12)	Tips & Hints, Points System, Achievements, Avatar, Progress, Feedback
Webonauts Internet Academy [16]	Web-based Point-and-Click Side-Scroller	Player ranks up their status by demonstrating smart and good behaviour. Teaches children how to be respectful and protect themselves online via website certificates, establishing privacy settings and not giving out passwords.	Children (aged 7-12)	Tips & Hints, Feedback, Avatar, Badges
Targeted Attack [17]	Web-based Point-and-Click	Player is a CEO in a choice-based simulation which teaches employees the correct responses and mitigations to cyber-attacks through trial and error.	Business Employees	Feedback, Challenge, Story
Keep Tradition Secure [18]	Web-based Point-and-Click	Player is quizzed with security-related questions to take down a cybercriminal; rewards are given to participants.	University Students	Tips & Hints, Feedback, Rewards
Cyber Awareness Challenge [19]	Training Simulation	Player is placed in a workplace environment which provides security guidance according to how they behave.	Business Employees	Tips & Hints, Points System, Feedback, Story
Cyberland [20]	Web-based Point-and-Click	Cyber Security Challenge UK hosts a variety of mini games (Cyberland), and competitions between schools, universities, businesses and government institutions.	Students (All Levels)	Tips & Hints, Story, Goals, Competition, Feedback
Hacknet [21]	Video Game Point-and-Click	Hacknet is a terminal-based hacking simulator. It teaches the player how to navigate networks, search for hidden files/folders, bypass authorisation, and general Unix commands.	Gamers	Story, Progress, Achievements, Feedback
Classcraft [22]	Web-based Point-and-Click	Classcraft incorporates gamification principles using project management software. Teachers set goals and challenges within a classroom which encourages teamwork between students.	Students (Lower Education)	Points System, Leader board, Goals, Avatar, Competition, Feedback, Incentives, Badges

Table 1: Analysis of Pre-Existing Cyber Security Games

2.5 An Analysis of Games Reviewed

Many of the educational games reviewed in Table 1 relied heavily on presenting facts and then subsequently quizzing the user. However, users can utilise common sense to rule out incorrect answers. This fails to invoke critical thinking and keeping the user engaged [23]. As a solution, gamified strategies should incorporate a variety of factual, conceptual, and procedural learning methodologies [23]. Conceptual understanding should be prioritised due to security-risks often being linked together in multi-faceted attacks [2].

Of all games reviewed and found, only Game of Threats [14] supported multiplayer which suggests there are a lack of educational, multiplayer games accessible online. According to [24], educational, online and multiplayer games are rare as it is difficult for scenarios to be built and managed as the player number scales. This paper recommends that scenarios are kept simple and designed for just 2-4 players as open world games tend to lose focus [24].

Lastly, every game that was accessible online was targeting students and employees; this suggests a lack in resources available to the public.

2.6 Related Work

The following section reviews literature related to this project's proposed solution of creating a multiplayer tabletop game.

2.6.1 Control-Alt-Hack

Control-Alt-Hack [12] is a physical tabletop card game. The core gameplay instructs players to control 'white hat' hacker archetypes to complete missions in which cyber security scenarios are exemplified.

From the feedback gathered in [12], of 450 students and 150 educators, it was shown that social engagement was the biggest factor that positively impacted security awareness among the participants. However, the main criticism identified was that there 'there was not enough educational value which specifically tested their knowledge of vulnerabilities and penetrations techniques' [12]. This is likely due to the lack of strategy in the game, in which a conceptual strategic environment could provide a means to testing knowledge.

Figure 1: Control-Alt-Hack - Sourced from [12]

2.6.2 OWASP Cornucopia Game

OWASP Cornucopia [25] is a tabletop card game in which a deck of cards is mapped from the OWASP Top Ten [6] and CAPEC software attack patterns. The Cornucopia card deck is used to supplement the design process of an agile, software development project. Each card is used to facilitate conversation about secure development. In a post review study [26], it was found that teams who discussed security guidelines whilst using the Cornucopia expressed more enthusiasm and enjoyment towards their work than teams who did not, thereby exemplifying the effectivity of gamifying a scenario.

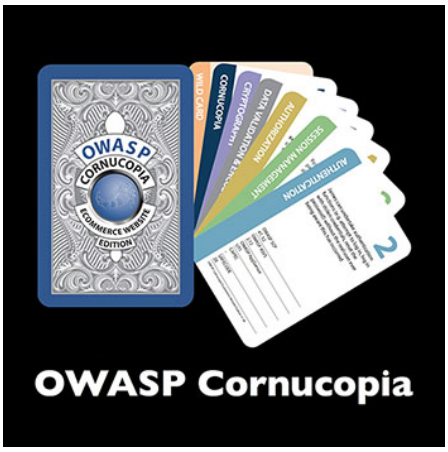


Figure 2: OWASP Cornucopia - Sourced from [25]

2.6.3 Program Wars

Program Wars [11] is a web-based 2-player game in which both players utilise a card deck related to programming principles. Because all participants were familiar with card games (and games of this nature), nearly all 46 participants had no difficulty playing the game without a fully implemented tutorial [11]. As such, this game was successful in enabling people to connect their conceptual learning, within the game-scenario, to real programming languages afterwards [11].

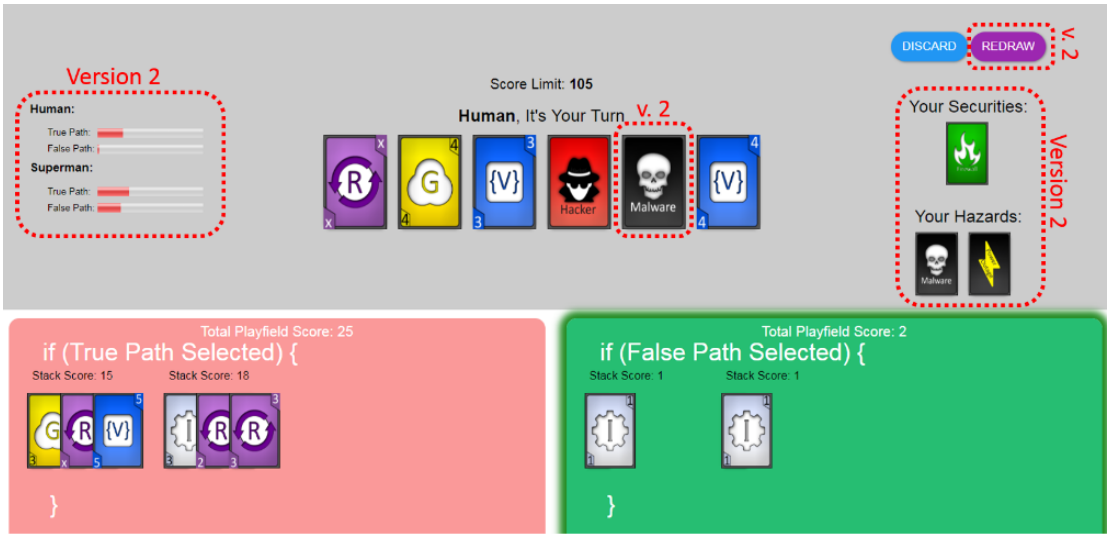


Figure 3: Program Wars - Sourced from [11]

2.6.4 Riskio

Riskio [9] is a 3-5 player tabletop card game. Riskio's core purpose is to elicit understanding in identifying threats within an office scenario. By using a combination of 'attack', 'information' and 'defence' cards, players are encouraged to identify with both sides of a cyber-attack scenario. Unlike the previous games, Riskio requires the role of a 'game master' with skilled cyber security knowledge to facilitate the learning environment.

In a post review analysis [9], this strategy was successful in eliciting a player's curiosity to read into each card to understand the core gameplay. However, it was also identified the importance of a cost function for user actions to create meaningful play [9].

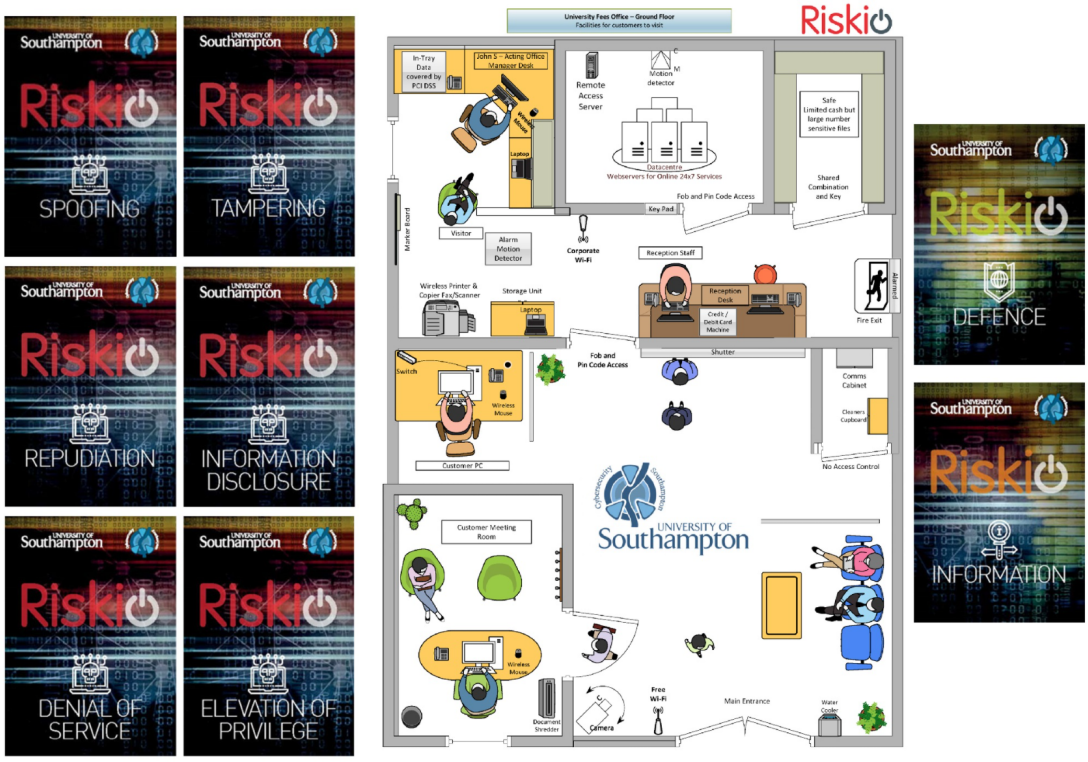


Figure 4: Riskio - Sourced from[9]

2.7 A Solution to the Problem Statement

From reviewing cyber security games, the following 8 features and gamified mechanics were identified as valuable components to make this project stand out, whilst building a solution to the problem statement.

Table 2 illustrates the comparison of these features to this project.

Feature / Mechanic

1. Multi-platform - Available on PC, Web and Mobile
2. Multiplayer - Supports online matchmaking
3. Competition - Between friends and colleagues
4. Educational - Incorporates an educational aspect within the game
5. Hints & Tips - Provided by the application
6. Accessible - Free and readily available online
7. Multiple Audiences - Relevant for all age categories and occupations
8. Extensible - Flexible to add new scenarios and 'gamify' a subject

Application vs Feature ID	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Game of Threats	✓	✓	✓	✓				✓
Cyber Security Labs	✓			✓	✓	✓		
Webonauts Internet Academy				✓	✓	✓		
Targeted Attack	✓			✓		✓		
Keep Tradition Secure	✓			✓	✓	✓		
Cyber Awareness Challenge				✓	✓	✓	✓	
Cyberland	✓		✓	✓	✓	✓	✓	✓
Hacknet				✓			✓	
Classcraft	✓		✓		✓	✓		✓
This Project	✓	✓	✓	✓	✓	✓	✓	✓

Table 2: Proposed Features for this Project

2.8 Inspiration from Related Work

Building on from the feedback identified in related work, the following conclusions have been implemented in this project:

Cost Function

Regarding [9], to prevent the core gameplay not eliciting meaningful play, applying a 'cost function' to the underlying mechanics should add an extra depth of strategy.

Familiarity

By developing a game-type that users are already familiar with, the application should be easy to understand and adopt [11]. As such the game will be inspired from a typical turn-based board game like Chess.

Implementing Strategy

As a major criticism in Control-Alt-Hack [12] was the lack of depth. Designing a game with a ruleset of mechanics like Chess, should invoke critical thinking whilst testing the user's acquired knowledge within the application.

2.9 Conclusion

To summarise, a cyber security game needs to be cost effective and target the specific needs and requirements of the workplace to be viable for SMBs. Where traditional training methodologies fall short, gamified strategies can offer a more cost-effective solution for teaching relevant cyber security concepts. However, many of the educational games reviewed were flawed from a lack of gamified mechanics that encourage meaningful play. Multiplayer, Challenge and Competition are some key mechanics which were missing from freely available cyber security games. Therefore, the goal is to incorporate these mechanics into an application in which scenarios can be flexibly created for enjoyment and training.

3 Requirements

In this section, personas are constructed from the stakeholders who could benefit from the proposed application. From this, requirements are identified and refined into user stories with a MoSCoW priority. These user stories are then scheduled into a Gantt Chart for project management.

3.1 Stakeholder - Personas

From the literature review, a list of stakeholders in which educational cyber security games were targeting includes:

- Students (All Levels)
- Researchers and Professors
- Employees (SMBs)
- Remote Workers
- Security Advocates

Persona One - Olivia - Student

Figure 5: Source [27]

Olivia is an undergraduate Physics student with a strong interest in web development. As such, she would like to learn more about the most common web application security risks.

Olivia also wishes to make her parents more cyber security aware as her own father has succumbed to phishing emails whilst working from home.

As her Dad suffers from colour blindness, she requires a colour-blind friendly game in which she could learn the rules and play with her father.

Olivia could benefit from the application by playing out multiple scenarios with her father. This would inform her about secure web-development, as well as engaging her father with the world of cyber-security.

Persona Two - Ben - Software Consultant

Figure 6: Source [28]

Ben is a software consultant for a medium-sized business. Although he has a degree in Computer Science, he is not an expert in cyber security.

Ben would like to develop software for his clients that is as secure as it is usable. Since this is done in a team, Ben would like to encourage his co-workers to participate in keeping up to date with current security trends.

His requirements are free, lightweight training material as he does not have the required funding, or time, to hire an expert security consultant.

To benefit his goal, the application will have competitive multiplayer that will encourage his employees to challenge each other in a stimulating way.

Persona Three - Samira - Security Analyst

Figure 7: Source [29]

Samira is a senior level security analyst who completed her PhD on using gamification to teach cyber security.

As an expert in her field, her passion for cyber security extends to all of her colleagues, friends, and family. She has a strong desire in raising awareness and eliminating the human error which lead to most security exploits.

Samira would benefit from the application by using it as a framework to map alternative cyber-risk scenarios, thus creating a training tool. Furthermore, she could use the application as a foundation to further her own research in gamification techniques.

3.2 Functional Requirements

The following requirements are necessary features that the application requires to reach a minimum viable product.

ID	Requirement	Description	MoSCoW
1	Multiplayer	Multiplayer turns between connected users	Must
2	Matchmaking	Users can host and join a game session	Must
3	Quit / Leave	Users can leave at any point	Must
4	Web Accessible	Playable on a web browser	Should
5	View Units	Users can see the status of all units	Should
6	View Description	Units and moves have an explanation	Must
7	View History	Users can see a history of events	Must
8	Complete	Users can play a full version of the game	Must
9	Clear Goal	Gameplay loop and objectives are clear	Must
10	Single Player	Users can play single player	Won't
11	Movement	Units move with varied distances correctly	Must
12	Unit behaviour	Units have unique attributes, moves and abilities	Must
13	Tutorial	Users can learn the rules and mechanics easily	Must
14	Gamification	Gamification mechanics should incite learning	Must
15	Challenge	The game should invoke critical thinking	Must

Table 3: Functional Requirements

3.3 Non-Functional Requirements

The following requirements outline how the application should perform.

ID	Requirement	Description	MoSCoW
16	Availability	Cross-platform playable (e.g., Tablet to Desktop)	Could
17	Ease of use	Easy to learn, understand and use	Should
18	Accessibility	Compliant with WCAG 2.1 and colour-blind friendly	Should
19	Secure	Playable on a secure website domain	Should
20	Correct	Factually correct in any cyber security concepts explored	Must
21	Performance	Gameplay should exceed 30 frames per second	Must
22	Scalable	Scalable to support more than 2 players at any given time	Should
23	Learning	Users can learn deeper cyber security principles	Must
24	Extensible	Flexibility to add different scenarios, units, and abilities	Won't
25	Feedback	Provides feedback based on performance (e.g., Chess)	Could
26	Music	Good music that fits the theme and aesthetic	Should
27	Animations	Unit movement and abilities have satisfying animations	Should
28	Enjoyable	Depth of mechanics & dynamics that create meaningful play	Must

Table 4: Non-Functional Requirements

3.4 User Stories

The following user stories are derived from the functional, and non-functional, requirements (linked by the **ref** column). They illustrate the specific features the application should have in accordance to their MoSCoW priority.

ID	As a (role)	I want a/to.. (feature)	So that I can... (benefit)	MoSCoW	Ref	Difficulty
1	User	View a unit's attributes	Know who to attack	Must	5,6	Low
2	User	View an ability's attributes	Know what I can do	Must	5,6	Low
3	User	View a colour-blind friendly UI	View the application easily	Should	18	Low
4	User	View a unit's movement range	Where I can move to	Must	8, 11	Medium
5	User	View a unit's ability attack range	Know who I can reach	Must	8, 11	Medium
6	User	Move a unit	Attack a target	Must	8,12	High
7	User	Attack a target	Kill an enemy unit	Must	8,9	Medium
8	User	Disable a target unit	Prevent them from moving	Must	12	Medium
9	User	Defend an ally unit	Prevent them from dying	Must	12	Medium
10	User	Click on a move description	Learn about the attack nature	Must	6	Low
11	User	Click on a unit	Learn about the attack vector	Must	7	Low
12	User	View a history log	Follow the history of events	Should	7, 17	Low
13	User	View the units remaining	Know what units can still move	Should	17	High
14	User	Pan the camera and/or zoom in	See the map from different angles	Could	5	Low
15	User	Unique animations for each ability	Differentiate abilities	Could	18	Medium
16	User	Varied unity abilities	Enjoy varied gameplay	Could	27	Very High
17	User	Cost requirement for each ability	Think deeply about my actions	Should	12	High
18	User	An intermediate tutorial	Understand the gameplay	Should	13	Very High
19	User	Hints and tips bar	Understand the gameplay	Should	13, 23	Medium
20	User	Multiplayer functionality	Play with a friend	Must	1, 2 ,22	Very High
21	User	Single-player functionality	Play solo	Won't	10	Very High
22	User	Resign and quit	Leave the session	Must	3, 8	Low
23	User	Win or lose	Finish a game session	Must	8, 9	Low
24	User	Play on Desktop	Play on my PC	Must	16	Medium
25	User	Play on Web Browser	Play on my browser	Should	4, 16	Medium
26	User	Play on Tablet	Play on my tablet	Should	16	Medium
27	Employee	Play cross-platform sessions	Organise fun, training sessions	Could	16, 18	Low
28	Student	Learn about the OWASP Top Ten	Understand attacks and mitigations	Must	21, 28	High
29	User	Purchase and upgrade new units	Have a reason to keep playing	Could	12, 24	Very High
30	Employee	Receive feedback after a game	Become cyber-aware	Could	23, 25	Very High
31	User	Hear gameplay music	Become immersed within the game	Should	26, 28	Low
32	Researcher	Create new scenarios	So I can train & inform my friends	Could	24	High

Figure 8: Identifying User Stories

3.5 Constraints and Limitations

- The application must be optimised to run on a web browser with weaker graphics hardware (e.g., mobile devices)
- WebGL 2.0 [30] has a varied level of support for different web browsers
- The application will require user interface rescaling for different browsers; two aspect ratios of 16:9 and 4:3 will be prioritised
- The game must be fully playable with one button mapping to support touch screen gameplay
- Given the development time limit, only one game scenario will be created
- Due to Covid19, it will not be possible to evaluate a physical paper prototype of the application
- It will not be likely to iteratively improve the game's mechanics from sufficient playtesting

4 Design

This section provides a top-level overview of the physical design, game-sequence, and the core components of the application.

4.1 Prototyping the Application

Figure 9 presents the initial prototype of the application. Attack and defence cards represent unit abilities which are explored in the following section.

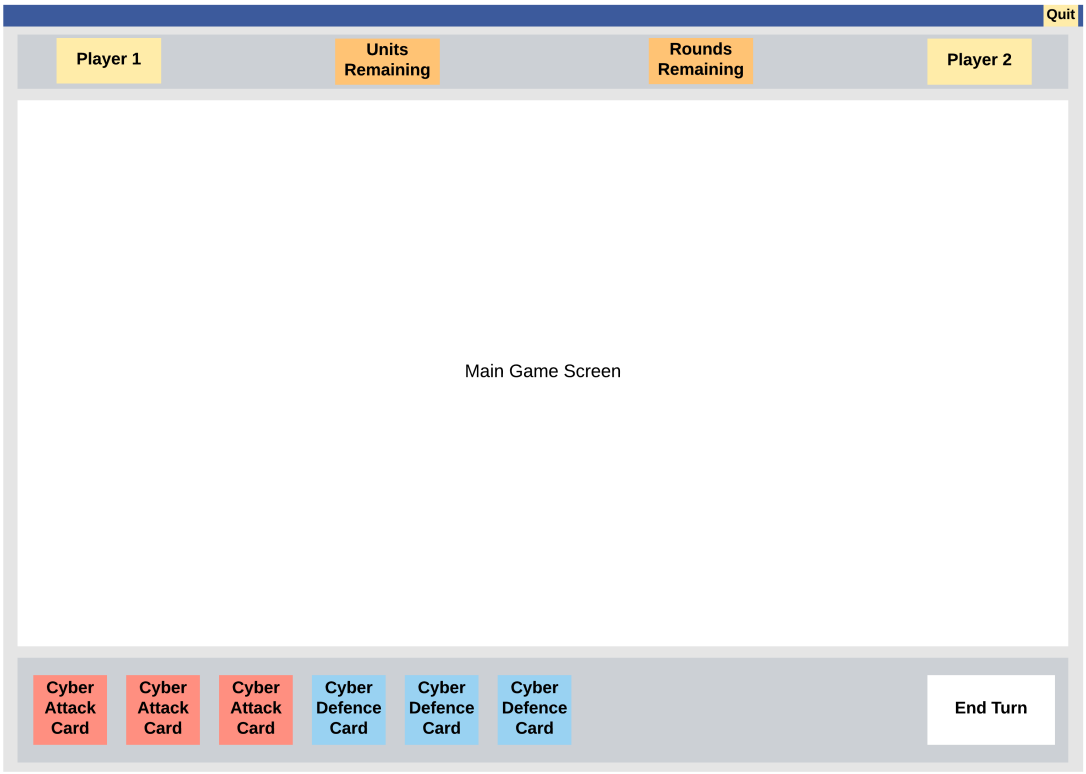


Figure 9: Wire-frame of Application

4.2 Applying the MDA Framework

The MDA (Mechanics, Dynamics, Aesthetics) framework [7] is the most well-known framework to identify the requirements of building a game. It is an iterative process in which the mechanics of the game are first outlined and then evaluated to see what game-play loops (dynamics) emerge. Finally, the dynamics are mapped to a target aesthetic e.g., the scarcity of resources is an important dynamic for a survival title.

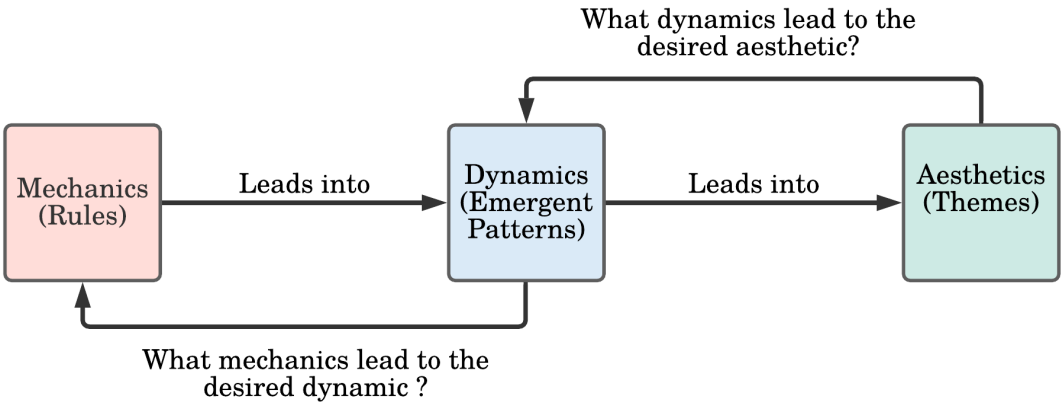


Figure 10: The Mechanics-Dynamics-Aesthetics Model [7]

4.2.1 Mechanics

The mechanics are the basic rules and actions a player can do within the game world [7].

They are as follows:

- Action Point (AP) represents a cost function for unit abilities
- Movement is blocked by enemy and friendly units
- Units can only attack and move once per turn
- Units cannot move after attacking
- Units have a maximum defence which cannot be restored past this threshold
- Health points (HP) cannot be restored
- When a unit reaches 0 HP, it will be removed from the game world
- Each movable unit has 2 abilities whereas the static units (database / web server) generate passive AP
- Abilities are limited by a distance range and can be blocked by other units
- Abilities can damage HP, defence, bypass shields, restore defence and disable other units
- Disabling a database / web server unit will halt AP gain for one turn
- Disabling movable units will prevent them from acting for one turn
- Abilities have unique effects versus certain units (e.g. SQL Injection deals critical damage to the database server)

4.2.2 Dynamics

The dynamics can be thought of as the 'emergent behaviour that arises from gameplay when the mechanics are implemented' [7].

They are as follows:

- Players may choose to target the web server/database to reduce AP gain
- Players may use shield destroying moves on heavy units
- Players may use cheaper moves to save AP
- Players may physically block units to protect them
- Players may use the bypass defence move to target units with a lot of defence but little HP (e.g. web server)
- Players may refer to the unit information to discover which moves have unique effects on certain units
- Players may position their units to benefit from, or avoid, multi-hitting movies

4.2.3 Aesthetics

The aesthetics can be thought of as the 'emotional response a game should illicit from the player' [7].

They can be broken down into:

- Sensation (emotion-invoking)
- Fantasy (immersion)
- Narrative (story - rich)
- **Challenge (puzzles / obstacles)***
- **Fellowship (co-operative / multiplayer)***
- Discovery (open world)
- Expression (character creation)
- Submission (simulations)

* Where Fellowship and Challenge are the two target Aesthetics in this application.

4.3 Mapping the OWASP Top Ten to Game Mechanics

Figure 11 represents the choice of mapping security risks, exploits and mitigations to in-game mechanics. A description and justification of these mechanics are explored in section 5.8. The OWASP Top Ten was chosen as it 'represents a broad consensus of the most common web application security risks' [6].

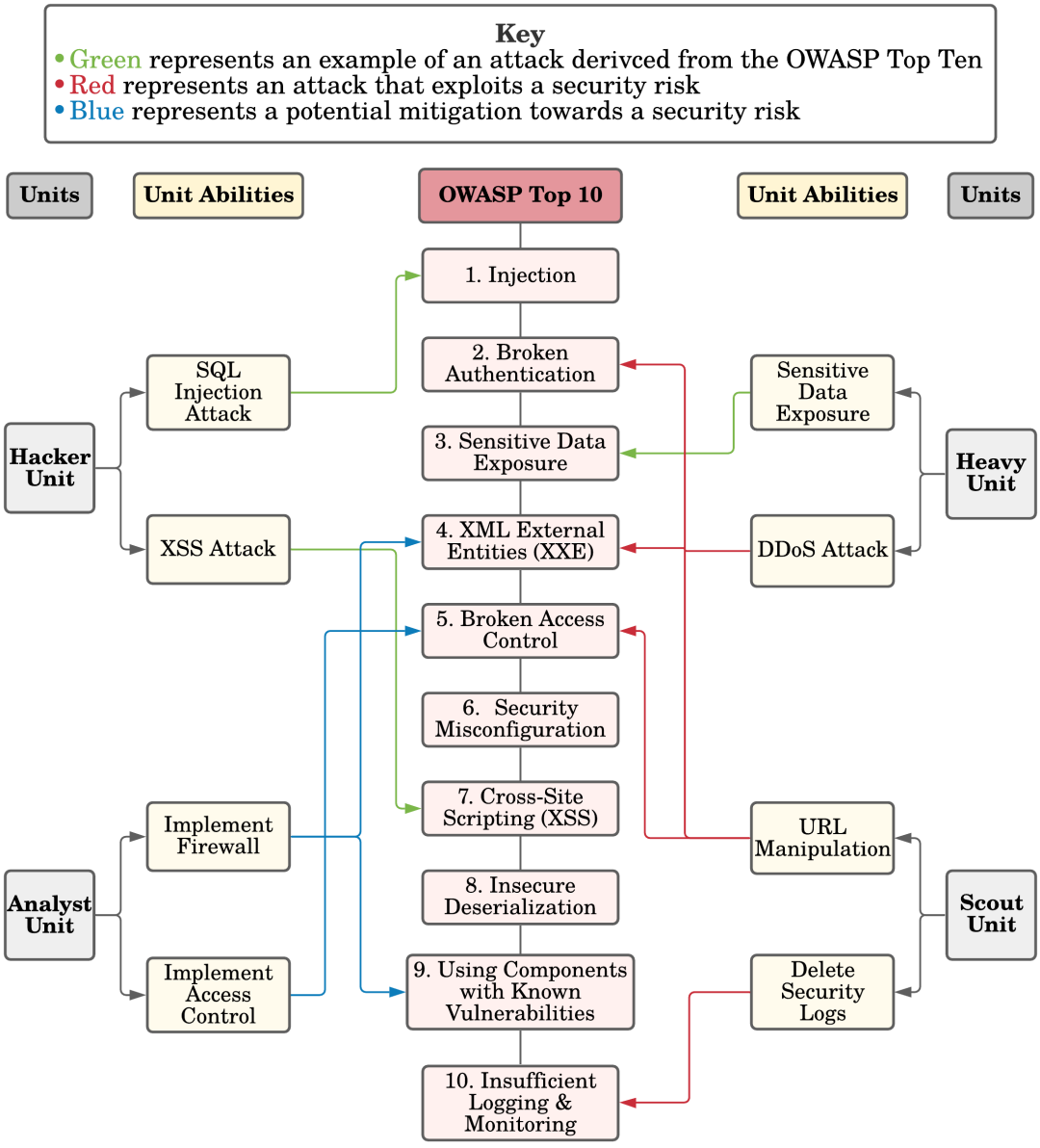


Figure 11: Mapping the OWASP Top Ten to Game Mechanics

4.4 UML Modelling - A Class Diagram Representation

Figure 12 represents the core components of code within the application but omits all testing, utility, and helper scripts.

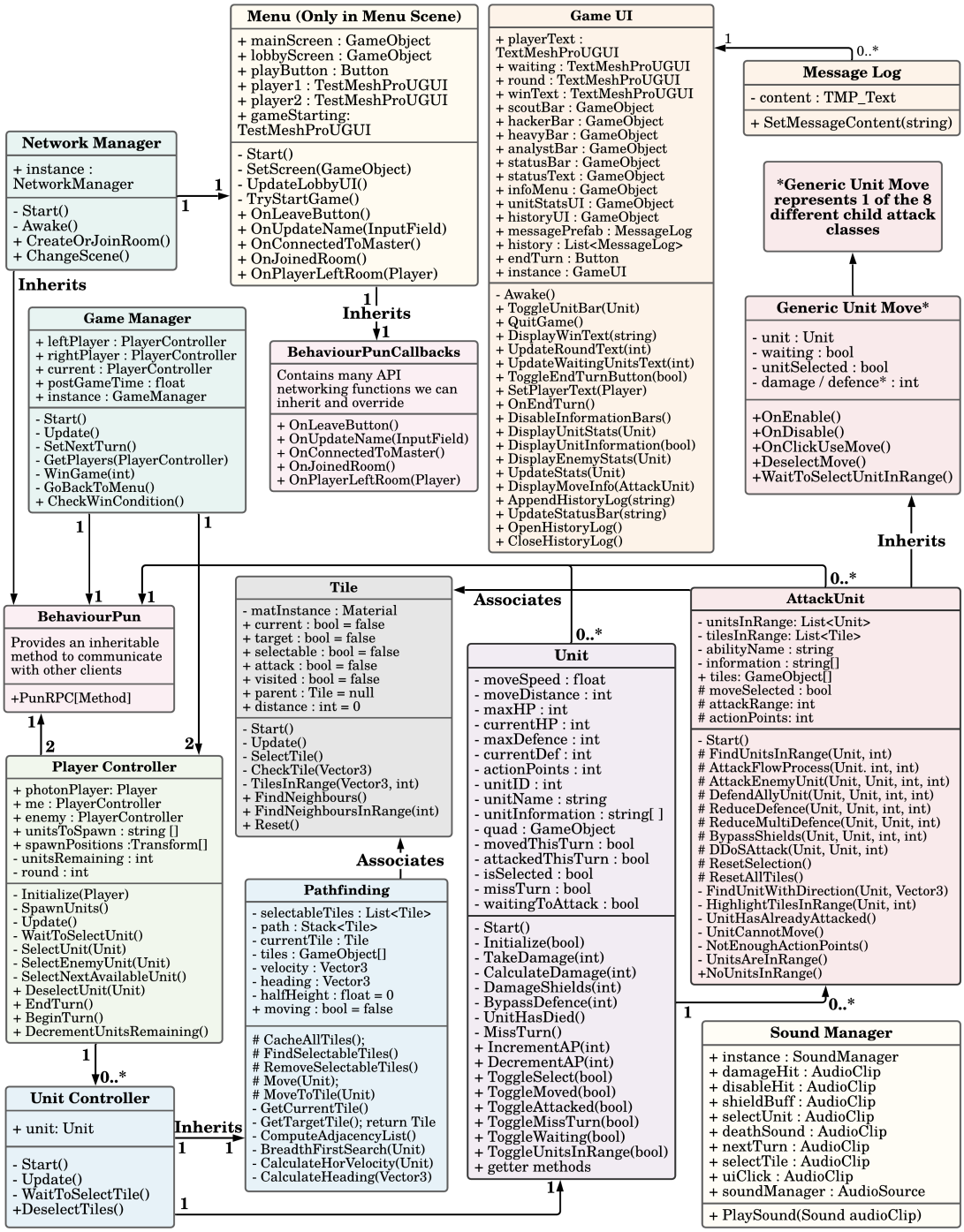


Figure 12: A Class Diagram Representation

4.5 Mapping the MDA Framework to Class Functions

Figure 13 maps components that relate to the MDA framework. Due to Aesthetics being an emotional product of the game, in this diagram it represents the physical assets and user interface that the player interacts with.

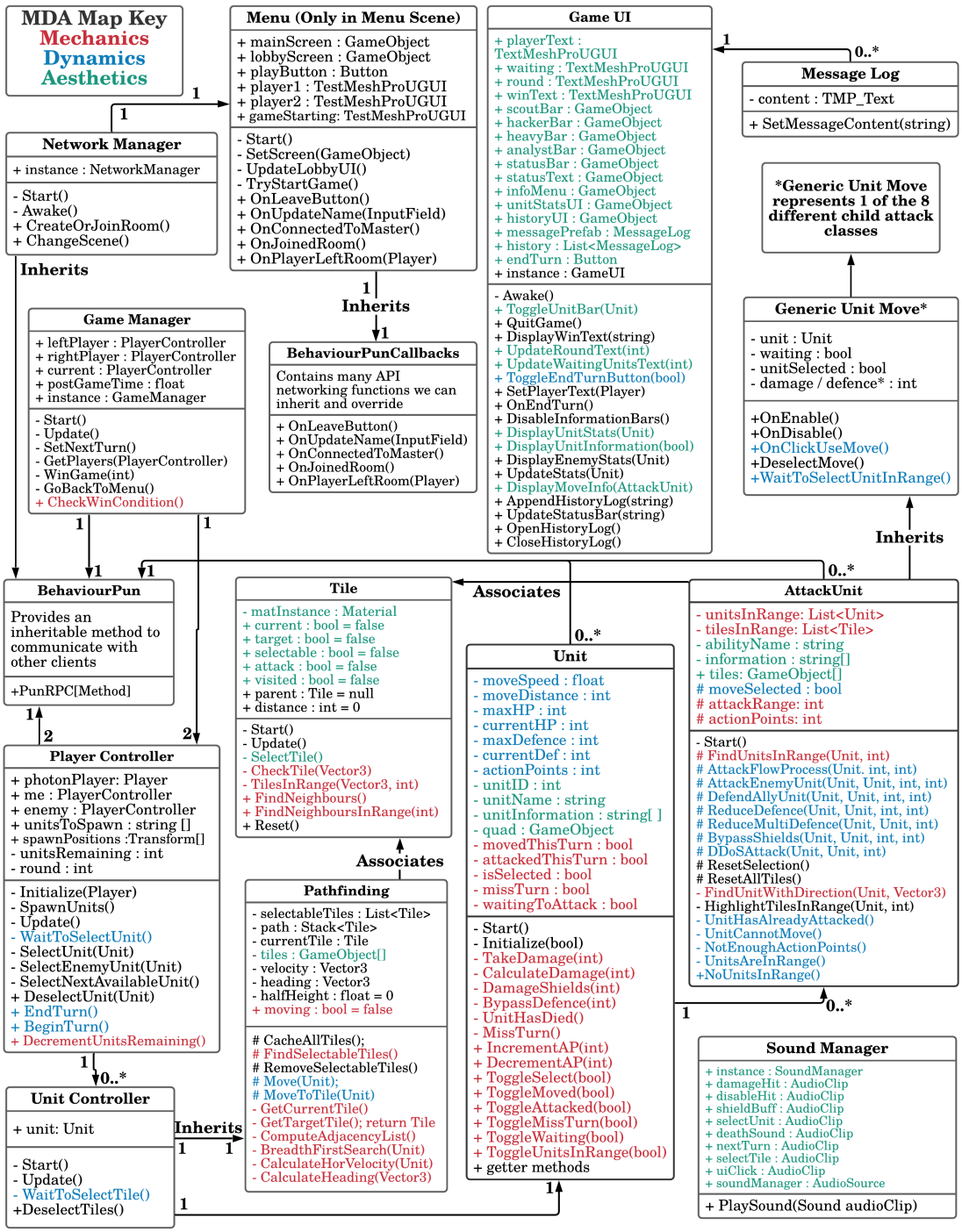


Figure 13: Mapping the MDA Framework to Class Functions

4.6 Description of Components

With reference to Figures 12 and 13, an explanation of the core classes and components are as follows:

Game Manager - a singleton instance of this class forms the backbone of the game as it controls the game's flow sequence, turn-manager, and game scene.

Behaviour PUN Callbacks - provides networking functions that enable multiplayer functionality. This class is provided by the Photon Unity Networking 2 API.

Network Manager - a singleton instance of this class inherits the PUN2 API to handle hosting, joining, and leaving a room.

Sound Manager - a singleton instance of this class provides functions to invoke sound effects.

Menu - awaits user input (from within the game menu) to invoke functions within the Network Manager.

Game UI - handles the transition and display of the user interface components.

Player Controller - listens for player input once per frame which allows unit selection. This also manages the player's turn and status of all units. A singleton instance of this class is instantiated for each player.

Unit Controller - enables tile selection and unit movement by inheriting functions provided from the Path-finding class. Each unit has an instance of this class.

Path-finding - utilises Breadth First Search to find a list of tiles that a selected unit can move to. It checks each tile for occupation, and highlights it green if it is a valid destination. The path-finding functions within this class were adapted from a tutorial available at Game Programming Academy [31].

Unit - contains unit information (e.g., health, defence), and functions relating to receiving damage, restoring defence and the unit's status. Each unit has an instance of this class.

Attack Unit - provides functions for finding units in range, attacking, and defending units. It associates with the Tile class to accomplish this.

Generic Unit Move - represents the 8-unit abilities which are derived from the OWASP Top Ten (e.g., SQL Injection). These scripts inherit from Attack Unit for shared functionality and are invoked by the user interface.

4.7 UML Modelling - Activity Diagram

Figure 14 demonstrates the typical sequence of two players joining a game, matchmaking and then the taking turns until one player wins.

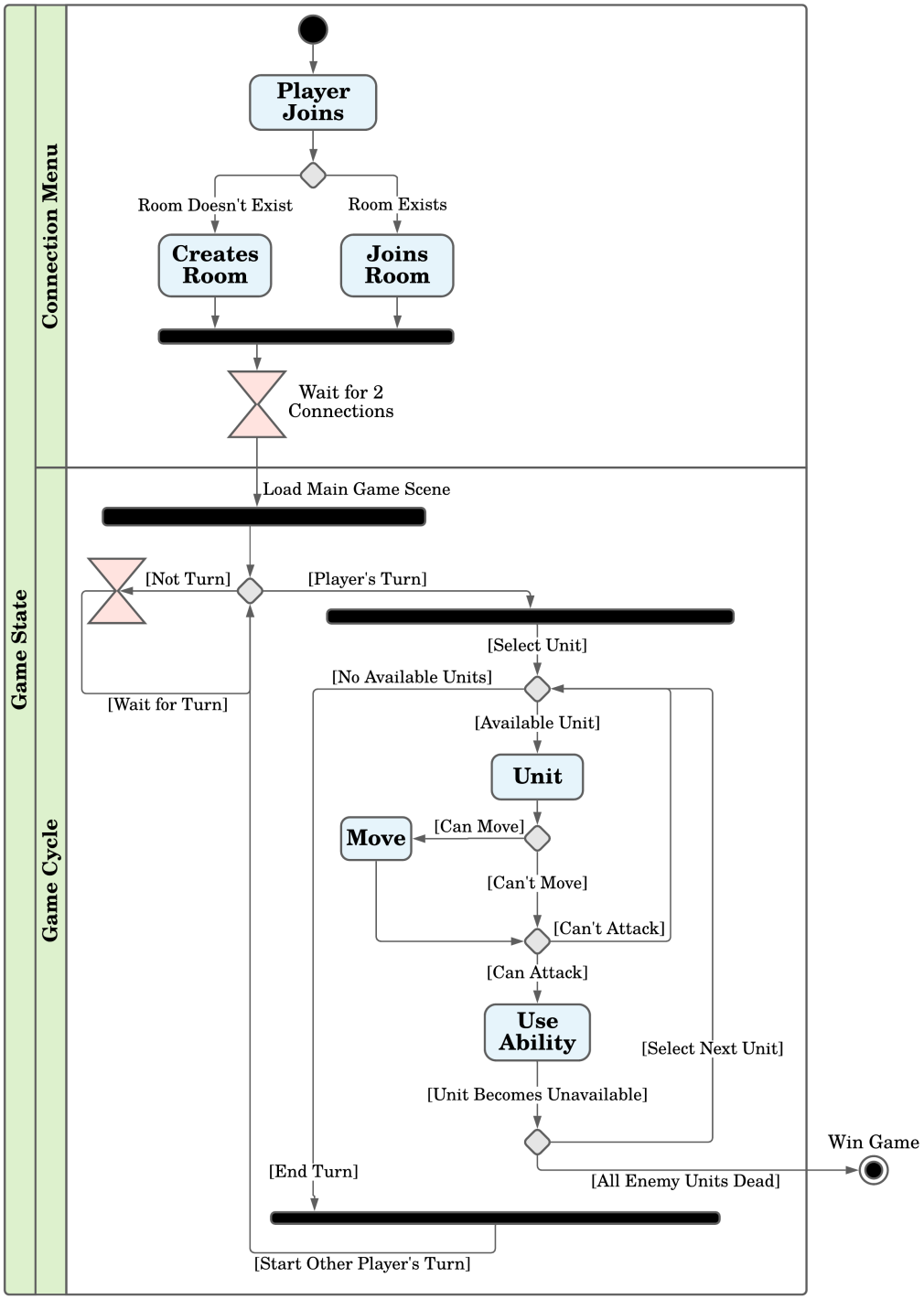


Figure 14: Activity Diagram of a Typical Match

4.8 Visual Design, Contrast and Accessibility

To facilitate an immersive game-like experience, the application needs to fit the theme of cyber security. As such, a colour palette was derived from NieR: Automata – a similarly themed title pivoted around machine intelligence. To ensure visual suitability for the web, the following complimentary and harmonic colours were incorporated into the background (Figure 15).

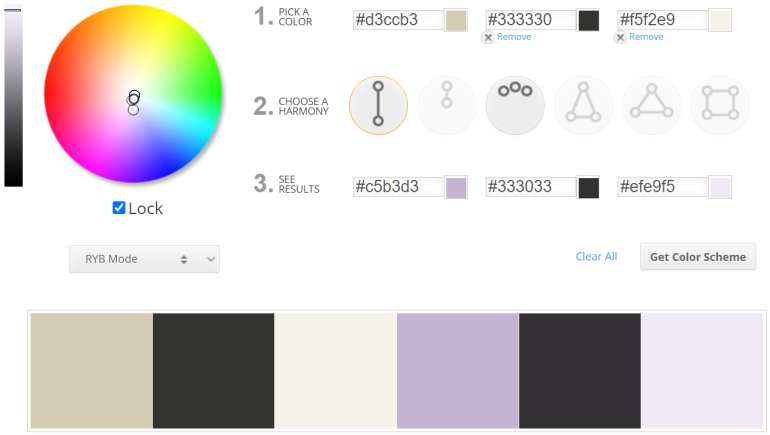
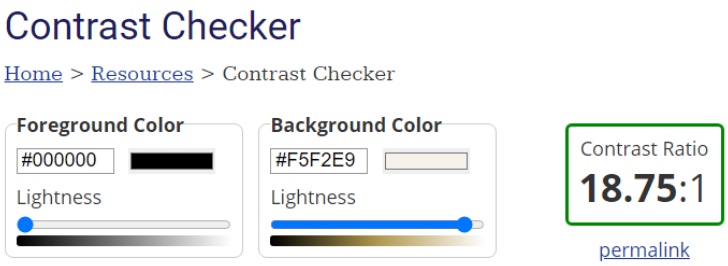
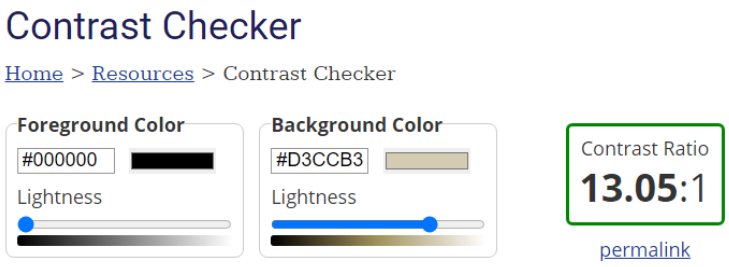


Figure 15: Identifying Complementary and Harmonic Colours - Source [32]

The Web Content Accessibility Guidelines (WCAG 2.1) requires that small text has a minimum contrast ratio of 7:1 [33]. All text used within the application has a contrast ratio of 18.75:1 and 13.05:1 which is compliant for all font-sizes.



(a) Larger Text - Source [34]



(b) Smaller Text - Source [35]

Figure 16: Contract Checking Text for Readability

4.9 Colour Blindness

Approximately 1 in 12 men, and 1 in 200 women, experience some form of Colour Vision Deficiency [36]; this exemplifies the importance of designing with accessibility in mind.

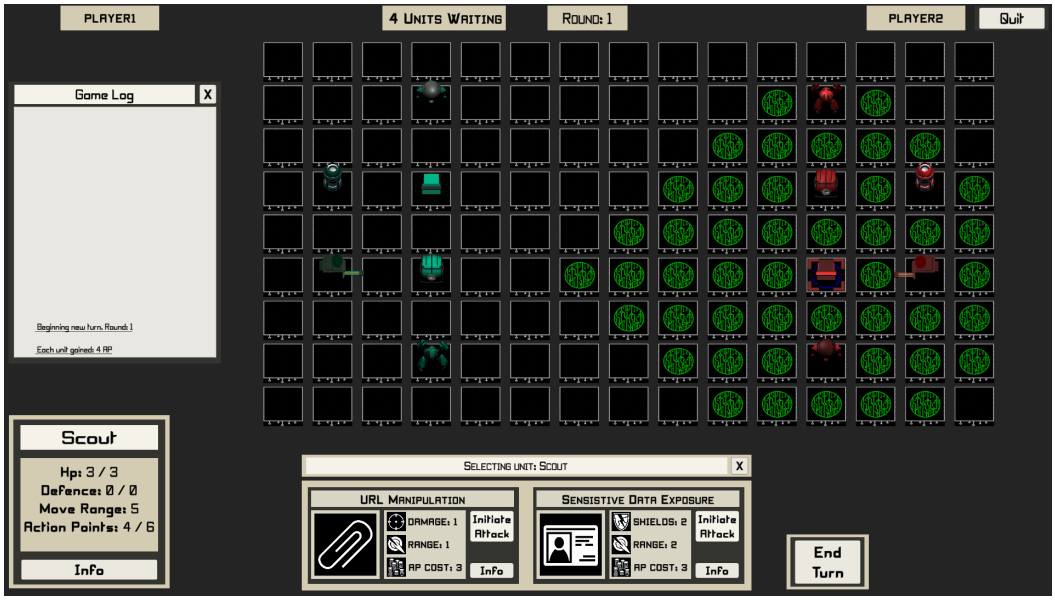


Figure 17: Initial Colour Scheme non-adhering to Accessibility Guidelines

Figure 17 presents the initial design and choice of unit-colour before amending these changes with a colour-blind friendly palette [37], and a background pattern [38] generated to better contrast the game-scene.

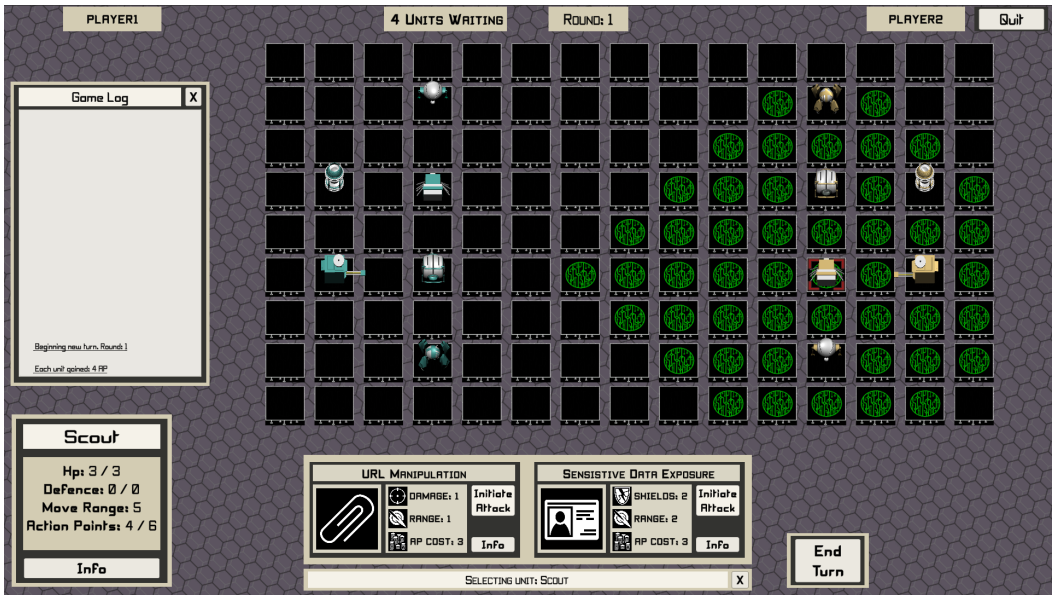


Figure 18: Revised Colour Scheme for the Desktop Application

4.10 Designing for the Web and Mobile

Due to technical limitations of rendering the tile design in WebGL, an alternative simpler build was developed. See Figure 19.

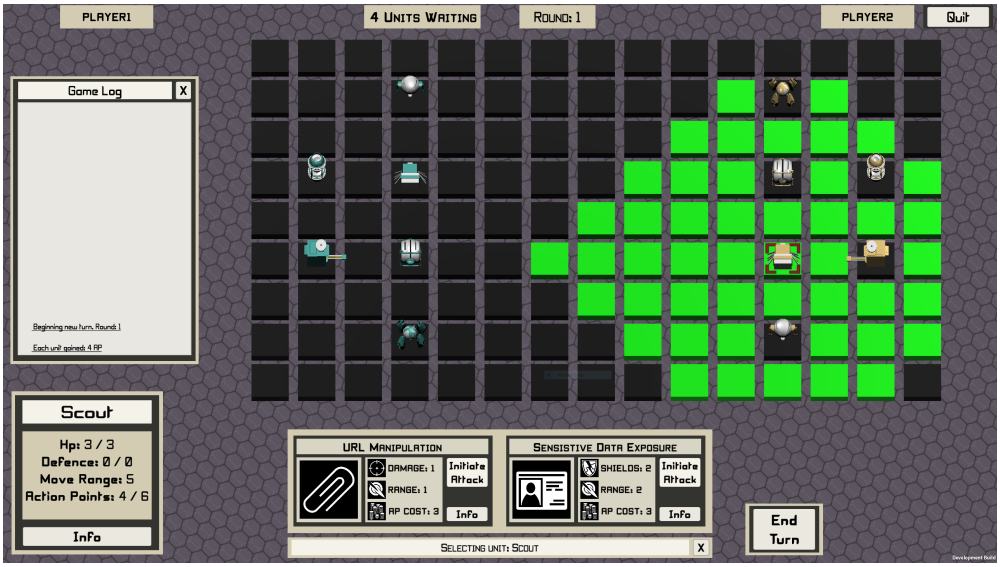


Figure 19: WebGL Build (16:9 Aspect Ratio)

To run on different devices, the application's user interface supports the standard monitor aspect ratio of 16:9, ultra-wide 21:9, as well as 4:3 for a tablet device. All controls support touch screen with a point-and-click button mapping.

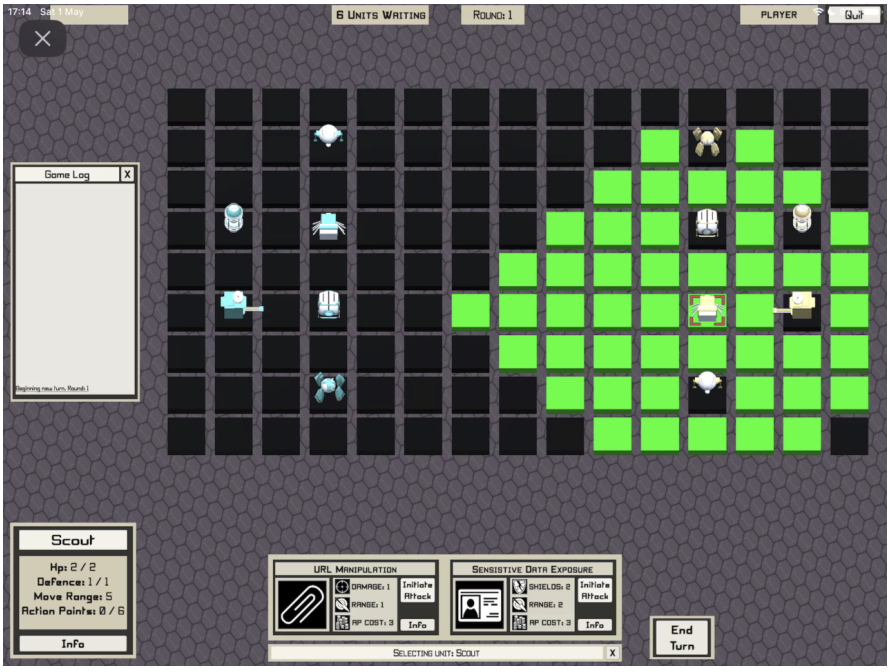


Figure 20: WebGL Mobile Build (4:3 Aspect Ratio)

4.11 Design Patterns for Unity & C#

The following section exemplifies the recommended design patterns for Unity.

4.11.1 Component Design Pattern

Unity is a component-driven engine, in which game objects, scripts and functions are broken down into smaller sub-components [39].

The 'Unit' game object exemplifies this as all unit functionality (attributes, movement, networking, rigid bodies and mesh colliders), are broken down into sub-components and stitched together (see Figure 21). Like object-orientated programming, this approach enables components to be decoupled, re-arranged, and re-used easily.

Furthermore, unit attributes, information and abilities are saved as prefabs which can be modified within the inspector freely. This allows for the creation of entirely new units and scenarios without any extra programming required.

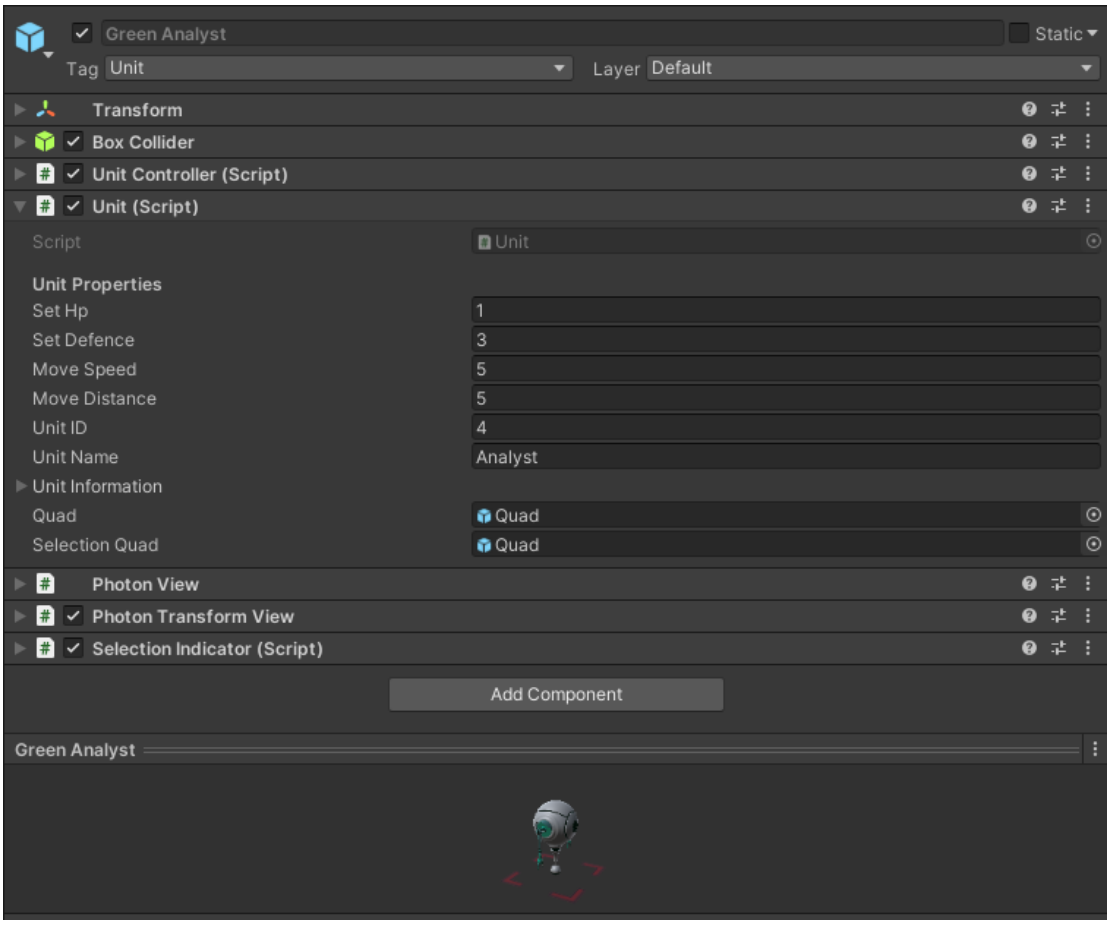


Figure 21: Example of Unit Sub-Components

4.11.2 Singleton Design Pattern

The singleton pattern involves creating a public static reference (of a non-static class) such that only one instance of that class exists [39]. This is useful for game managers as it is not possible to instantiate static classes. However, having a single, static reference is versatile in that it stores important information about the state of the game, and can be publicly invoked by other components without duplicate instances.

```
public class GameManager : MonoBehaviourPun {

    // Singleton instance of the Game Manager
    public static GameManager instance;

    // Assign instance upon initialisation
    private void Awake() => instance = this;
 }
```

Figure 22: Applying the Singleton Pattern

4.11.3 Flyweight Design Pattern

The flyweight pattern is designed to save memory by re-using instances of the same object [39]. For example, each time a tile changes colour, if a material instance is not saved and re-used, a new material object would be created which leads to unnecessary memory losses.

This is important because as the size of the game scales (with hundreds of tiles), lag spikes will occur more frequently as the garbage collector must be invoked to remove old material instances.

```
private Material _matInstance;

// Cache material component on initialisation
private void Start() {
    _matInstance = GetComponent<Renderer>().material;
}

// Re-use material instance each time the tile is updated
public void MarkSelectable() {
    selectable = true;
    _matInstance.color = Color.green;
}
```

Figure 23: Applying the Flyweight Pattern

5 Implementation

Unity [40] was chosen as the development environment because it provides functionality for exporting to a vast range of platforms – including WebGL [41]. Phaser3 [42] was considered as an alternative, web-development based solution. However, Phaser3 is a relatively young open source project whereas Unity has the largest online community, ample resources, and integration for supporting frameworks.

5.1 Multiplayer with Photon Cloud

Photon Cloud provides a software as a service solution to host cloud servers free up until 20 concurrent players [43]. For more scalable solutions, Photon Cloud requires paid services.

Photon Cloud is supported by Photon Unity Networking 2 (PUN2) [44], an API which enables developers to integrate multiplayer functionality into their games. PUN2 was chosen as it has strong integration with Unity and supports cross-platform networking.

Mirror [45] was considered as an alternative multiplayer library which was built to replace Unity's previously deprecated library (UNET). Although Mirror is completely free to use, relatively simple and highly covered, it does not provide cloud services for externally hosting a game server. As this would require running a server constantly, this is would not be a viable long-term solution.

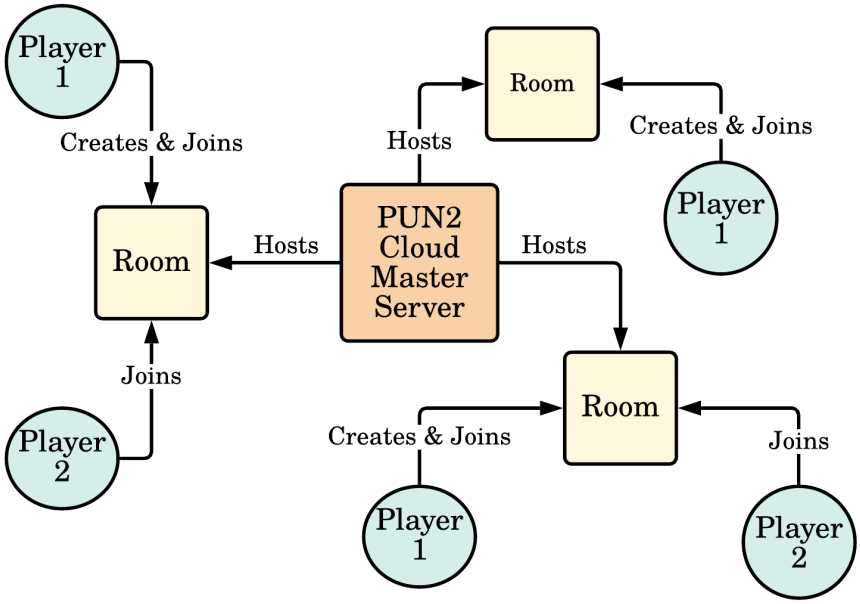


Figure 24: PUN2 Architectural Design

5.2 Photon Unity Networking 2 Architecture

PUN2 follows a classic server - client model (see Figure 24). When a client connects to the Photon network, it invokes the '*JoinOrCreateRoom()*' function which connects the client to an non-full room (if it exists), or creates one if there are no rooms available.

The first client that creates a room becomes the 'master' client which is responsible for hosting the game. If the master client disconnects for any reason, the next available client (if present) is promoted to the new master client. However, since this project is designed for two players, it returns the other player to the main menu and ends the room session.

5.3 Remote Procedure Calls (RPCs)

PUN RPCs [46] enable client - client communication in which programmed functions can be invoked on the opponent's client remotely. As both clients are running different, independent images of the game, RPCs are necessary to maintain concurrency by matching these images and state variables. For example, after a unit has attacked another unit, it will remotely call "Take Damage" to invoke this function on the recipient client.

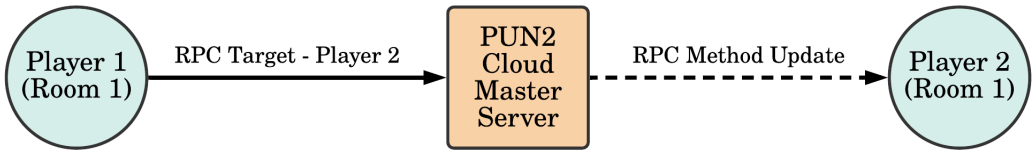


Figure 25: RPC Communication to the Other Client

By calling '*RPC Target.All*', the function is executed on all connected clients - including the origin client. This is the most often case as many multiplayer functions are state-changing (such as beginning a new turn and editing a unit's variables).

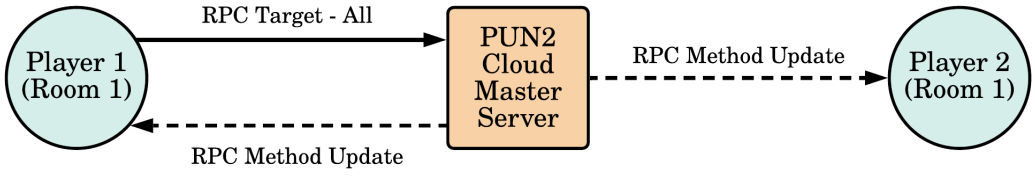


Figure 26: RPC Communication to All Users

5.4 Opening the Application

Figures 27 and 28 present the welcoming scene which enables auto matchmaking. In an earlier iteration, a lobby search system was created for 4-player games, however this was reduced as it's not necessary for 2-player games.

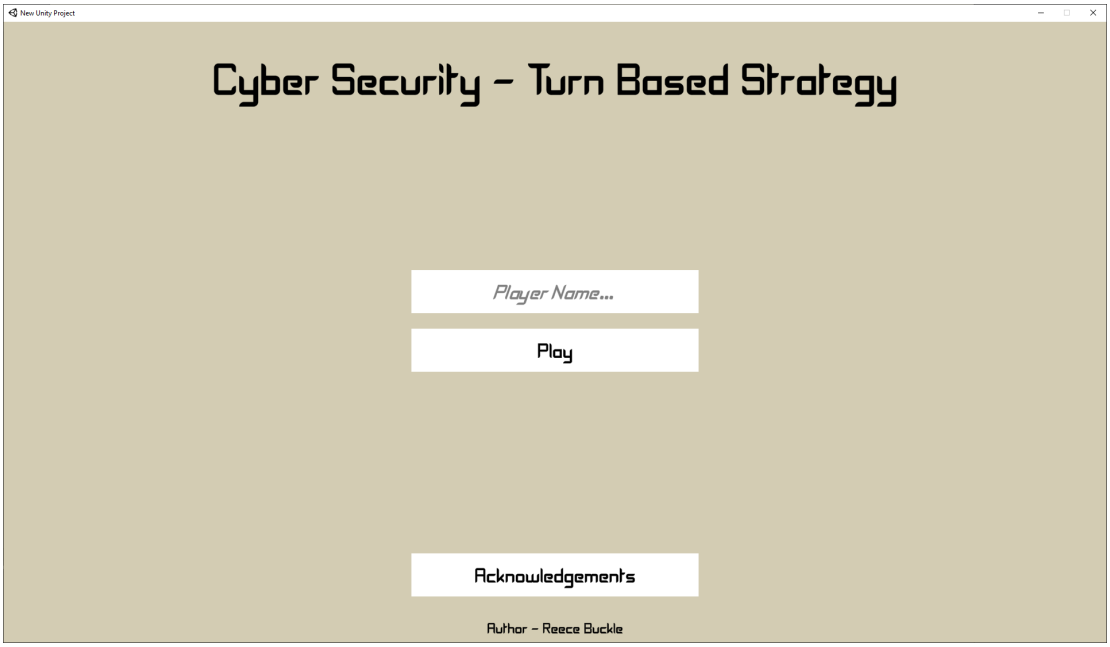


Figure 27: Welcome Menu

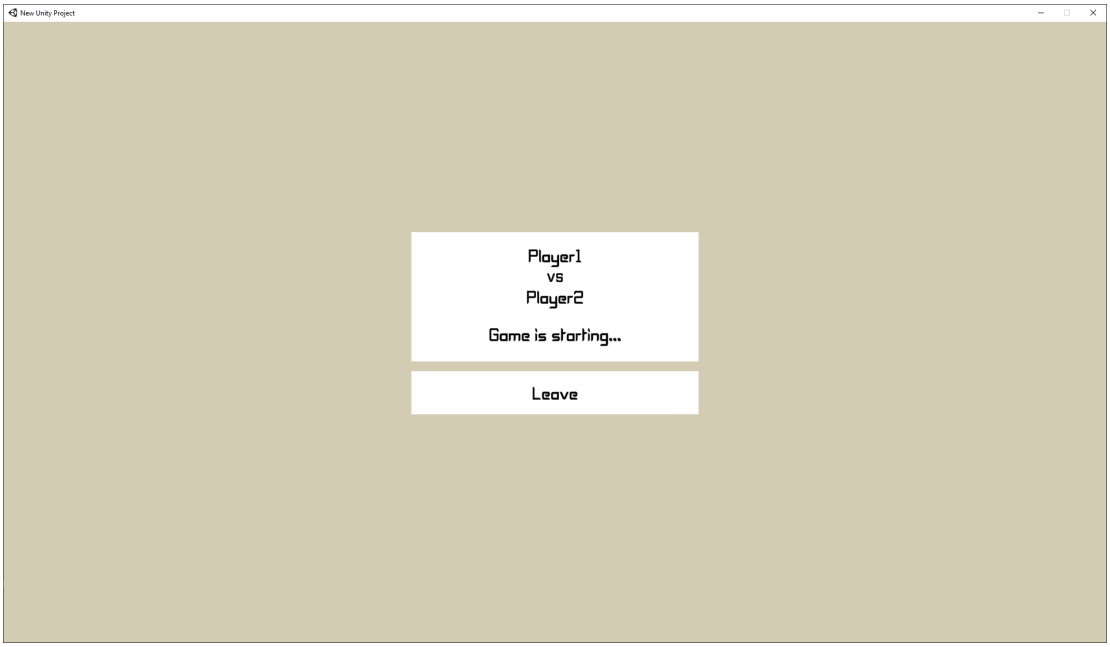


Figure 28: Connection Screen

5.5 Implementation of Unit Movement

A unit's movement distance parameter can be freely changed for any unit by setting its movement range within the inspector. Figures 29 and 30 demonstrate the range of tiles a selected unit can move to. Note these are blocked by other units.

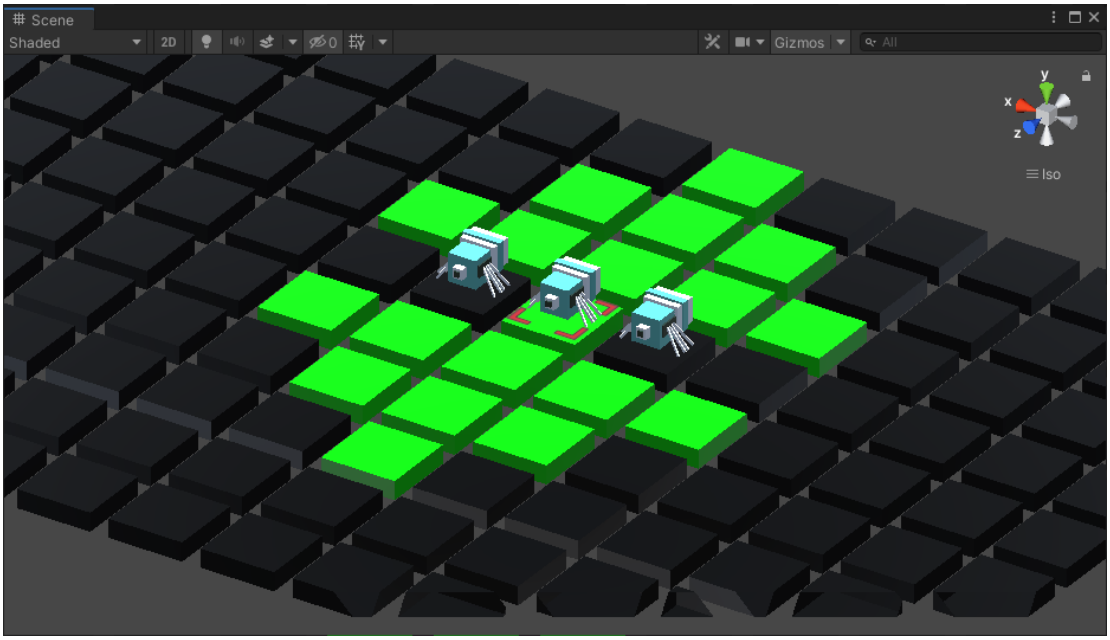


Figure 29: Movement Options for a Unit with Range: 3

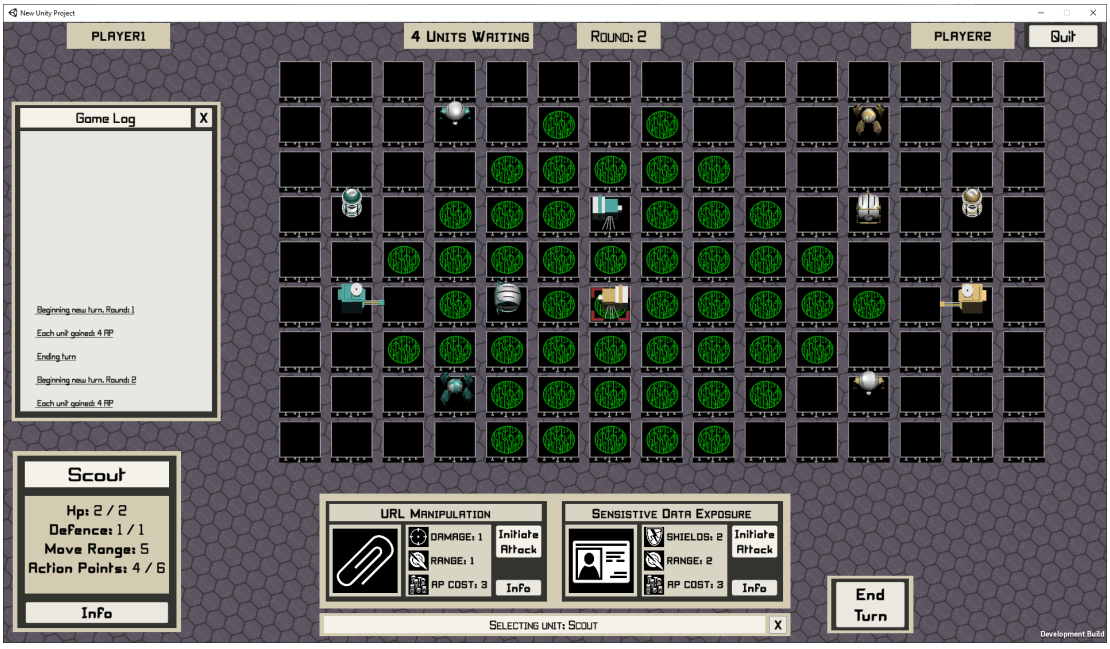


Figure 30: Movement Options for a Scout with Range: 5

5.6 Movement Mechanics - Breadth First Search

To calculate the selectable tiles in range of a unit, Breadth First Search (BFS) was implemented as it always returns the shortest path to the destination.

When a unit is selected, a ray-cast is sent below to find the tile underneath. From this tile, four more ray-casts are emitted north, east, south, and west to discover any neighbours. If these neighbours are walk-able, they are added to an adjacency list attached to that tile. Occupied tiles are ignored.

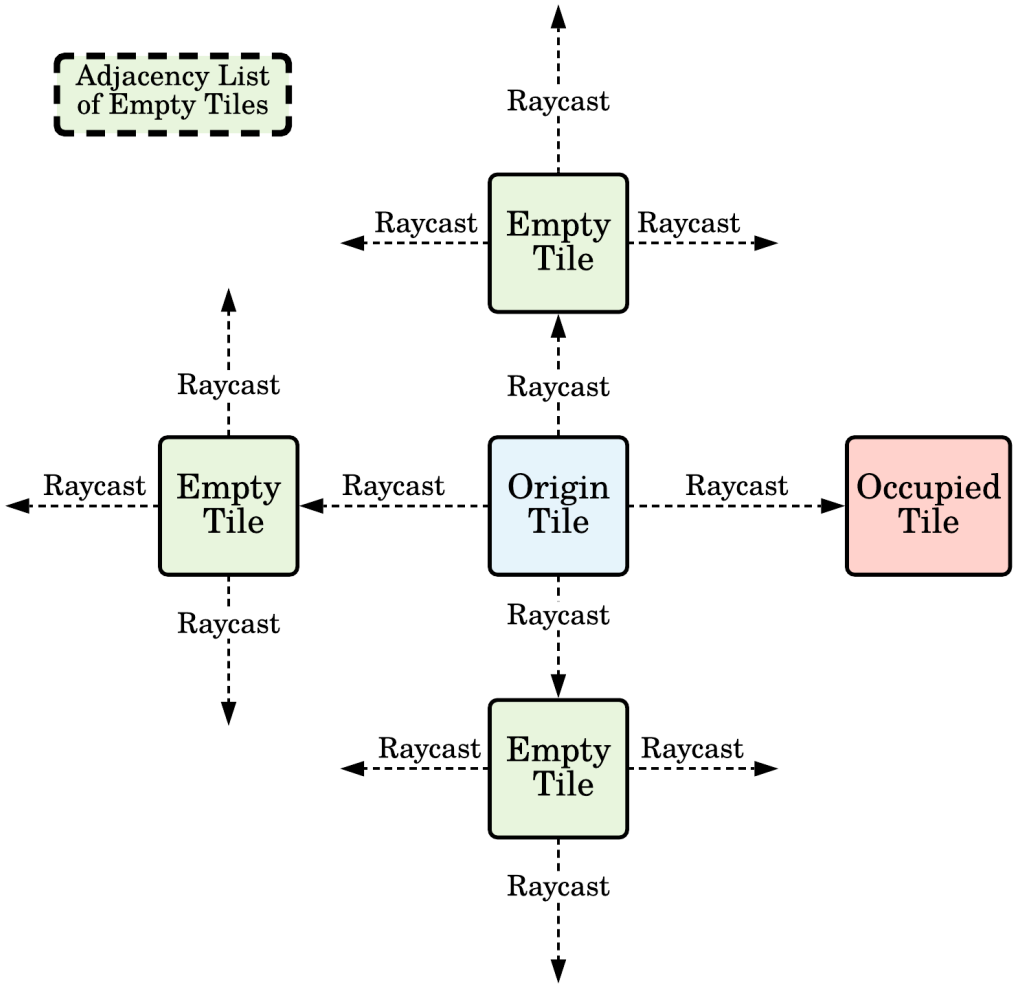


Figure 31: Ray-casting to Adjacent Tiles

BFS utilises a queue data structure. The origin tile is enqueued first. This is then dequeued such that it is marked as selectable, and all adjacent (walkable) tiles are processed. Each tile is marked as visited so that these tiles are not checked twice. See Figure 32 for a diagrammatic representation, and Figure 34 for the implementation.

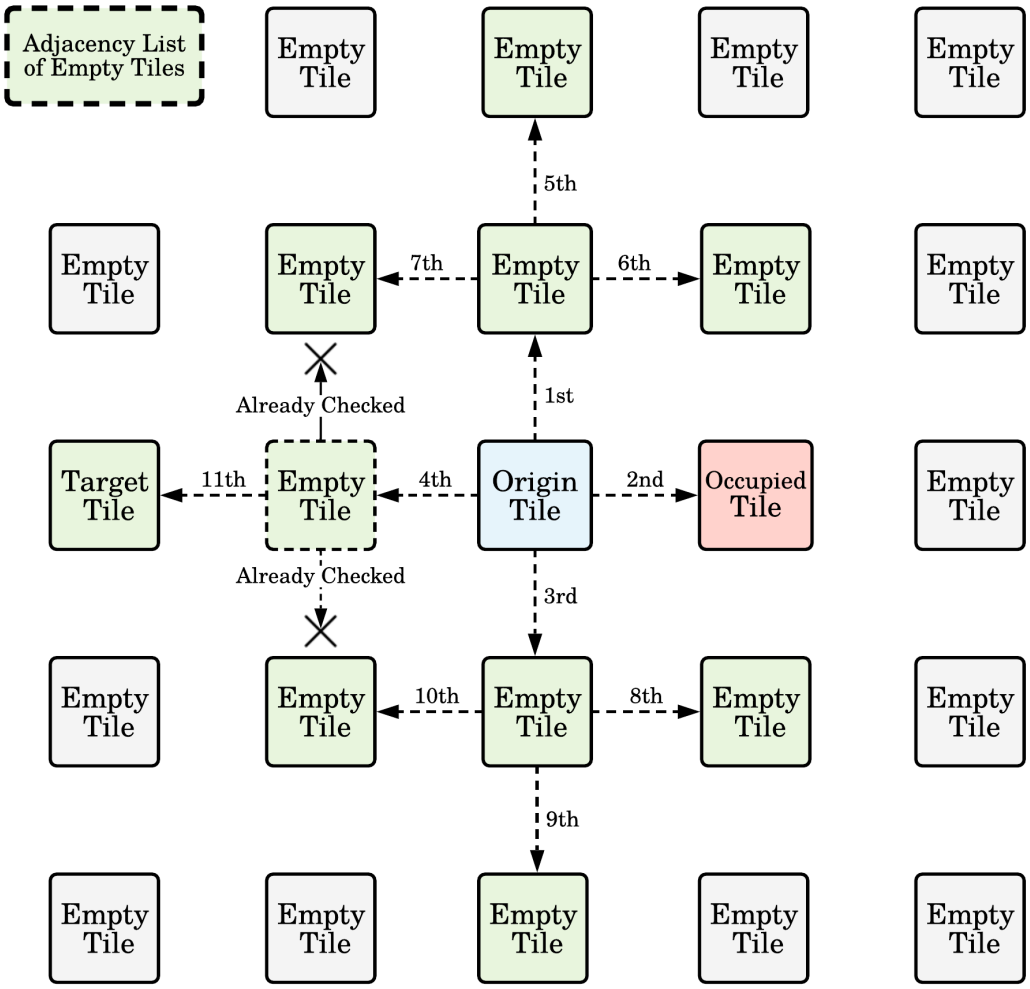


Figure 32: Finding Selectable Tiles via Breadth First Search

As each adjacent tile is processed, its parent tile is assigned to it. When a target tile is selected, its parent is recursively accessed until we arrive back at the origin tile; this always returns the shortest path to the destination.

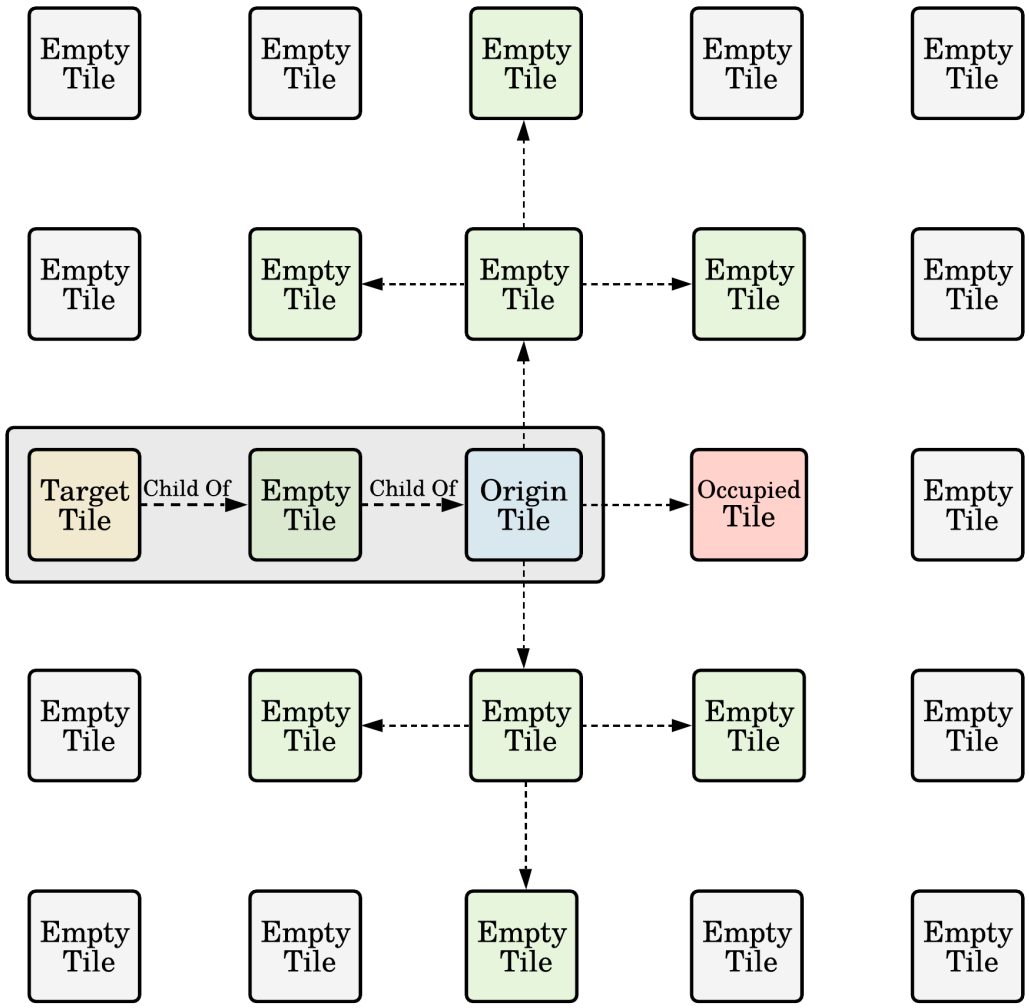


Figure 33: Finding the Shortest Path to Target Tile

Figure 34 describes the programming implementation of Breadth First Search.

```
private void BreadthFirstSearch(Unit unit) {

    Queue<Tile> tiles = new Queue<Tile>();

    // Enqueue tile underneath selected unit
    tiles.Enqueue(currentTile);

    // Mark as visited so tile is not processed twice
    currentTile.visited = true;

    // Begin dequeuing whilst enqueuing adjacent tiles
    while (tiles.Count > 0) {

        // Dequeue the first tile and mark as selectable
        Tile tile = tiles.Dequeue();
        selectableTiles.Add(tile);
        tile.MarkSelectable();

        // If tile is outside of range, check next tile in
           the queue
        if (tile.distance >= unit.GetMovementDistance())
           continue;

        // Search the 4 adjacent tiles
        foreach (Tile t in tile.adjacencyList) {

            // If tile has been visited, check next tile in
               adjacency list
            if (t.visited) continue;

            t.parent = tile;
            t.visited = true;

            // Update distance of tile from unit to keep
               track of how far we are from the starting tile
            t.distance = 1 + tile.distance;

            // Enqueue tile from adjacency list
            tiles.Enqueue(t);
        }
    }
}
```

Figure 34: Implementing Breadth First Search

5.7 Ability Range

Calculating an ability's range also utilises ray-casts to find units and tiles in range. However, this does not utilise BFS as it works in a line-of-sight manor. When a unit is selected, a target selection box is displayed underneath for a visual cue. Ability attributes can be set freely within the inspector.

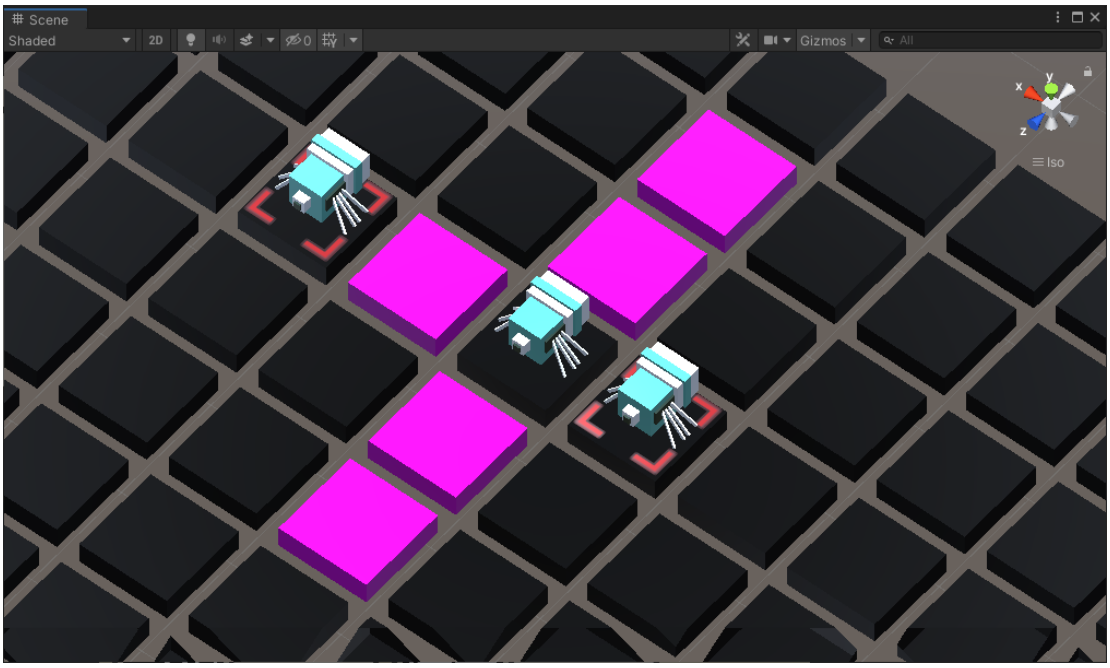


Figure 35: An Ability with 2 Range

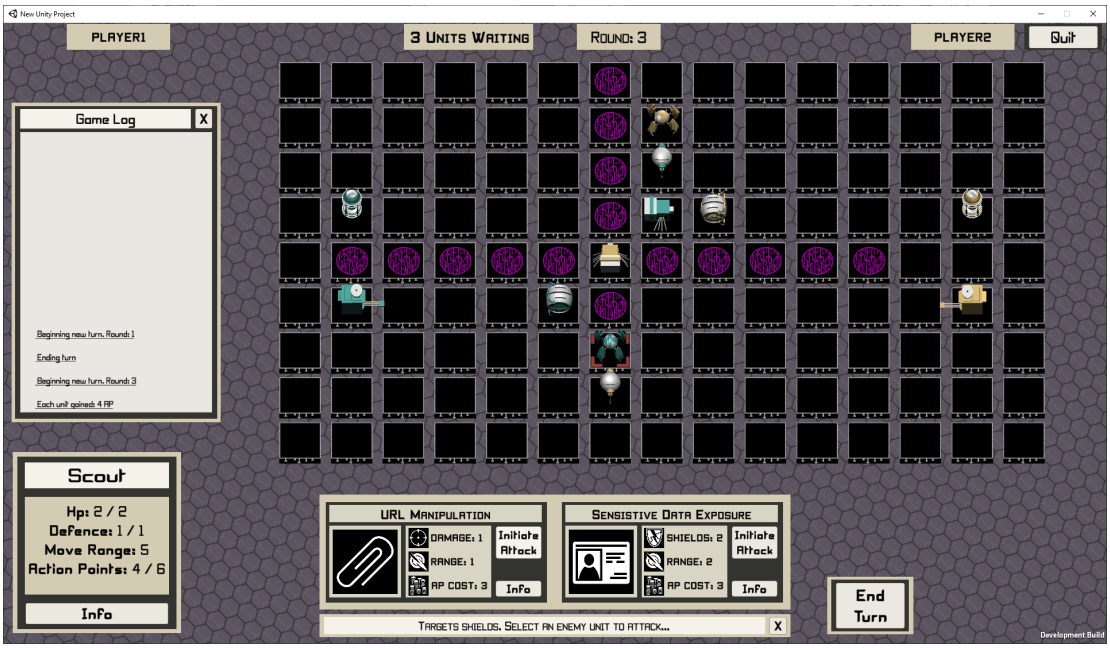


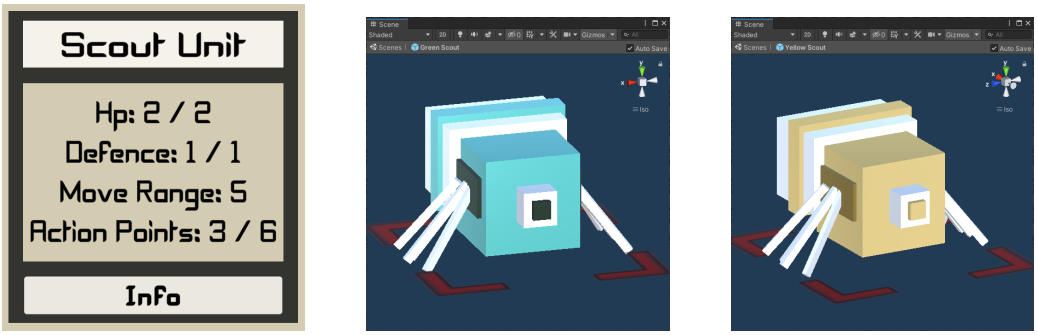
Figure 36: An Ability with 5 Range

5.8 Implementation of Units and Unit Abilities

The following sections describe the playable units and their abilities which are derived from the OWASP Top Ten [6]. Each description is loosely adapted from the information presented in the application. All model assets are royalty - free and accessible on Blendswap [47]. All ability icons are royalty - free images sourced from Game-icons [48].

5.8.1 Implementation of Scout Unit

The Scout unit represents exploiting broken control and exposing sensitive information from poorly designed web applications.



(a) Scout Stats (b) Green Scout (c) Yellow Scout

Figure 37: Scout Unit Stats and Model - Source [47]

An example of **URL Manipulation** involves changing the URL parameters to bypass authorisation control. Therefore, this move bypasses the opponent's defence in damage calculations. This move is close-range as it requires the unit to directly interact with the web application.

Sensitive Data Exposure includes leaking personal, account and financial information [49]. Improper handling of sensitive data in transit or storage could lead to data breaches which can cripple a company's reputation. In more severe outcomes, such information could be used to commit fraud and identity theft. As such, this move is a highly damaging defence-reducing attack.

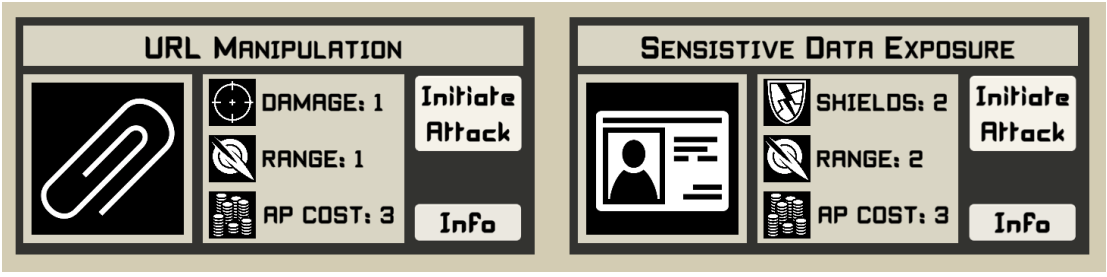
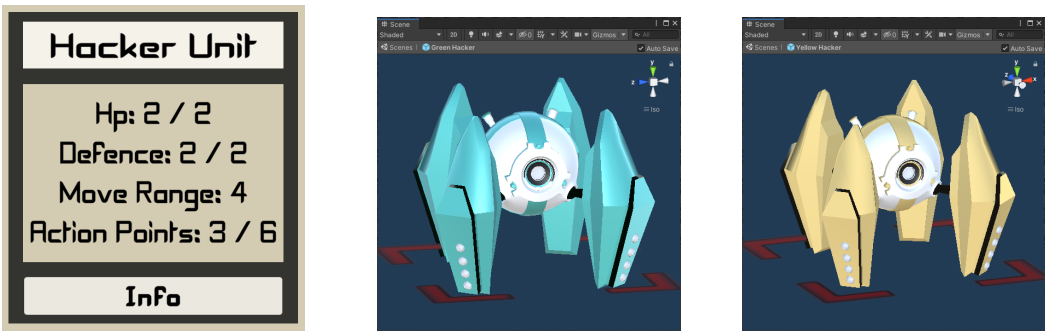


Figure 38: Scout Abilities - URL Manipulation and Sensitive Data Exposure

5.8.2 Implementation of Hacker Unit

The Hacker unit specialises with direct attack vectors typical of penetration testers who can prod a web application in search of vulnerabilities. SQL injection and XSS scripting are the two most common website security attacks, with XSS attacks occupying approximately 40% of all cyber-attacks [50].



(a) Heavy Stats (b) Green Hacker (c) Yellow Hacker

Figure 39: Hacker Unit Stats and Model - Source [51]

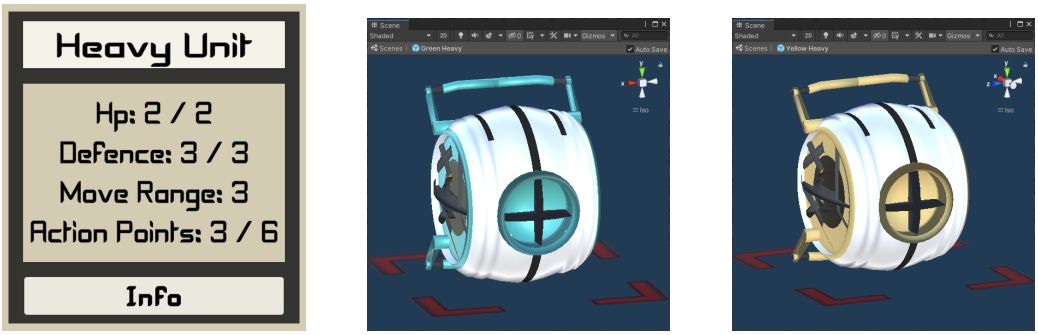
Cross-Site Scripting (XSS) attacks can occur when a web application fails to validate user input by escaping special characters [6]. As a result, an attacker can execute scripts in a victim's browser, hijack their session data and even redirect them to other phishing sites [6]. Therefore, this move is close-range and does +1 critical damage to the web server.

Sequel Query Language (SQL) injection exploits the interactivity between the web application and commands that query the back-end database [6]. Using SQL injection, it is possible to bypass authorisation, extract sensitive information from databases and even manipulate database entries [49]. Therefore, this move is long-range and does +1 critical damage to the database server.

Figure 40: Hacker Abilities - SQL Injection and Cross-Site Scripting Attack

5.8.3 Implementation of Heavy Unit

The Heavy unit represents physical disruption by disabling operations externally and internally.



(a) Scout Stats (b) Green Heavy (c) Yellow Heavy

Figure 41: Heavy Unit Stats and Model - Source [52]

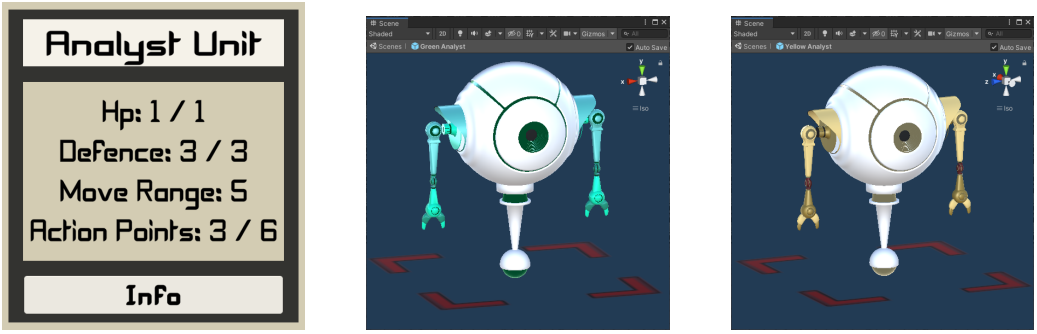
A **Distributed Denial of Service (DDoS)** attack uses a network of devices to flood a web application, network, or server with overwhelming internet traffic - and disable it from being able to handle requests [50]. As such, this move will disable any movable unit for 1 turn. If used on a server unit, it will halt action point gain for one turn.

Delete Security Logs exploits Insufficient Logging & Monitoring [6]. Examples, which include failing to record anomalies and login attempts, allow an attacker to persist in a compromised system without being detected [49]. As this affects the entire system, this move is multi-targeting and lowers the defence of all units in range.

Figure 42: Heavy Abilities - DDoS Attack and Delete Security Logs

5.8.4 Implementation of Analyst Unit

The analyst represents the role of security analysts who specialise in defending a network. This includes implementing security firewalls, keeping systems up to date and monitoring network activity.



(a) Analyst Stats (b) Green Analyst (c) Yellow Analyst

Figure 43: Analyst Unit Stats and Model - Source [53]

A **Firewall** is required to filter HTTP traffic and keep web-application components safely isolated. A web application firewall (WAF) utilises rules to mitigate common attacks such as XSS and SQL Injection [54]. Therefore, this move restores the shields of one unit in range.

Implement Access Control mitigates Broken Access Control [6]. A good access control system should be implemented once, and periodically reviewed [55]. Control policies based on user roles should enforce the least amount of privileges each user requires [55]. As this relates to all users, this move is multi-targeting and restores the defence of all units in range.

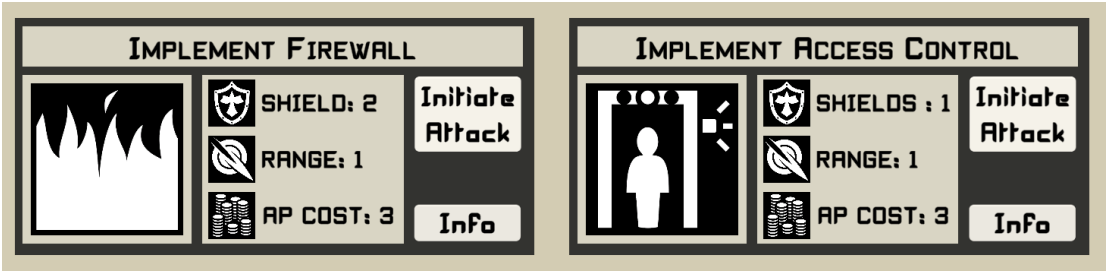
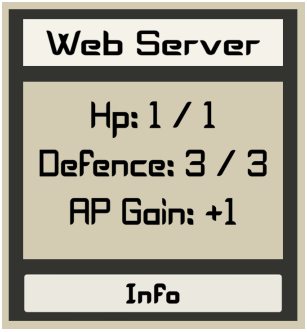


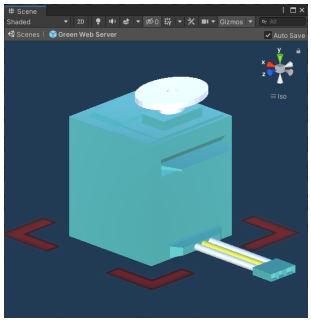
Figure 44: Analyst Abilities - Implement Firewall and Access Control

5.8.5 Implementation of Server Units

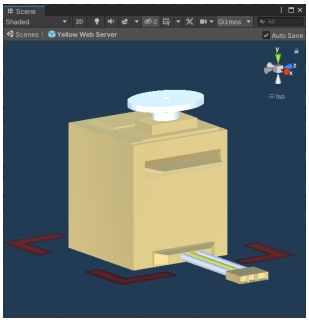
The web server is responsible for displaying web pages and handling HTTP network requests from all users accessing the web application. Therefore, it needs to be able to handle all user requests correctly whilst being robust to security risks [56].



(a) Web Server Stats



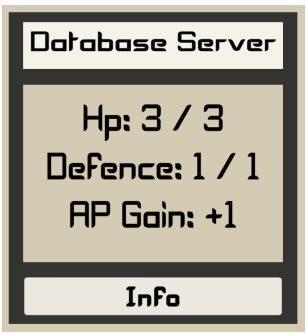
(b) Green Web Server



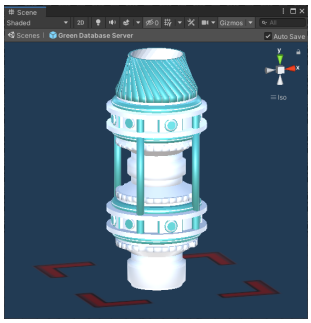
(c) Yellow Web Server

Figure 45: Web Server Stats and Model - Source [57]

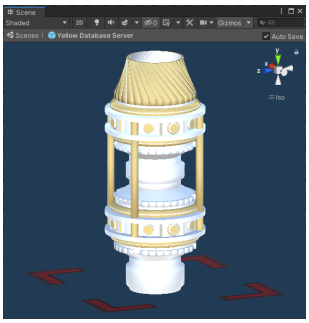
The database server manages back-end operations of the web application, and allows users to store and retrieve data. Important sensitive information should be kept isolated on a different server such that it cannot be accessed from front-end users [56].



(a) Database Server Stats



(b) Green Database Server



(c) Yellow Database Server

Figure 46: Database Server Stats and Model - Source [58]

Both server units generate +1 action points per turn to all other units, unless disabled or destroyed.

5.9 Information Menu

To promote learning, each unit or ability has an "info" button which provides information about an ability or unit's nature. This is recommended as it provides hints about which units an attack vector is extra potent against.

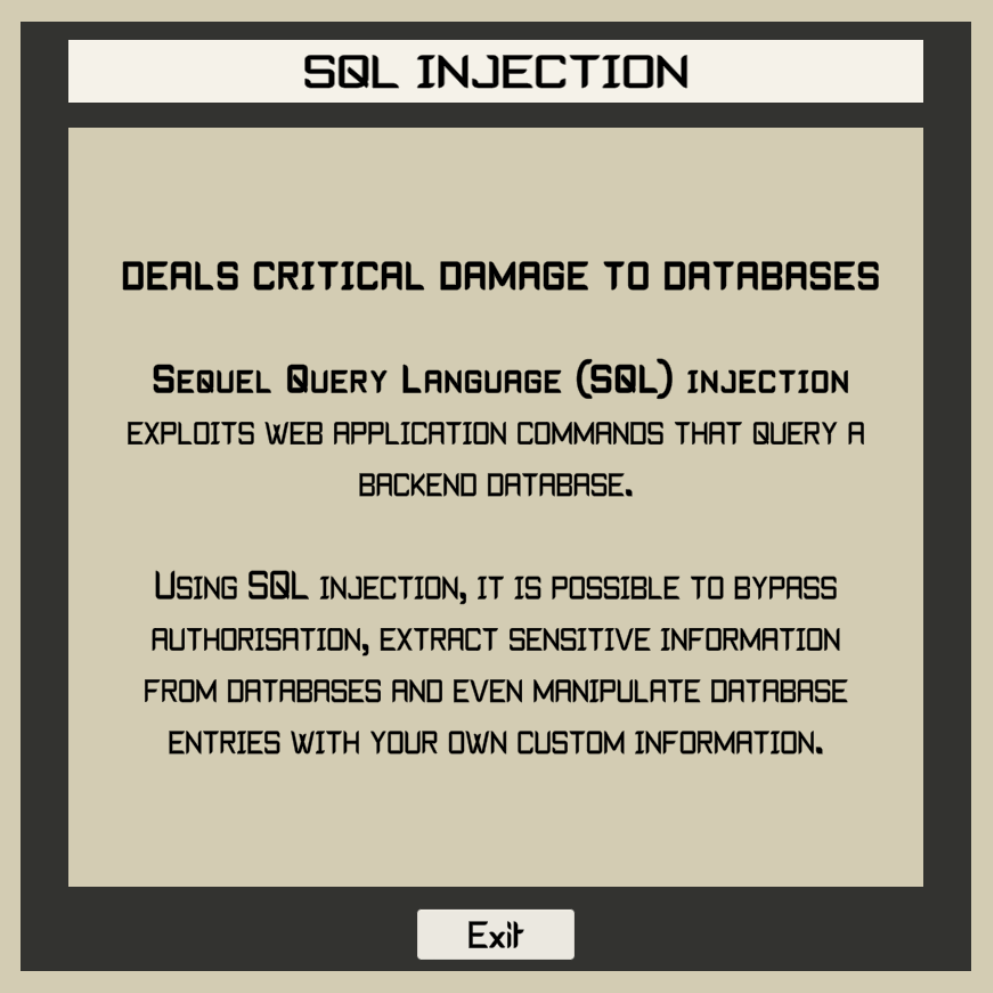


Figure 47: Information Screen Example - SQL Injection

5.10 Hints & Tips Bar

Figure 48 presents a tips & hints bar which summarises each mechanic in the game, and provides hints about the nature of each ability.

Figure 48: Selecting 'Implement Firewall'

5.11 History Log

Figure 49 presents a history log designed to facilitate ease of use and understanding. Each action and outcome in the game is appended to the history log of both players in their perspective.

Figure 49: History Log

5.12 Quit Menu

Figure 50 provides the player with the option to resign and safely disconnect to the main connection menu.

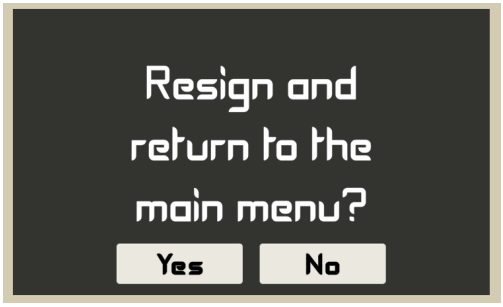


Figure 50: Quit Menu

6 Testing and Evaluation

This section explores the testing strategies implemented which include playtesting, unit testing, identifying edges cases and evaluating the overall performance with the Unity profiler. During programming, JetBrains Rider was used as an integrated development environment for its inbuilt functionality of providing performance recommendations.

6.1 Unit Testing

For small games, playtesting in Unity is quick and simple process. However, as a project expands, it can become very time-consuming to build the game, load menus, connect to another player, and perform actions to reach a desired game state and check one thing.

Figure 51 showcases automated tests that were performed on all pure functions within the application (i.e., functions that should always return the same output on a given input). These include all damage calculations and changing the state of selectable units.

Figure 51: Unity Test Runner

6.2 Identifying Edge Cases

To test these functions, all edge cases for negative, positive and zero values were identified and tested. A benefit of test-driven development is that it enforces good programming practises in that functions only perform one operation, can be debugged easily and flexibly re-written (without breaking other components). These tests are covered in Appendix 10.1.

6.3 Integration Testing with ParallelSync

ParallelSync [59] is an open source extension which clones the Unity development environment. This is necessary for integration testing a multi-client game as you cannot have duplications of the same environment open concurrently. Debugging would otherwise require rebuilding the project externally which can take several minutes per WebGL build.

ParallelSync works by cloning the folder structure of the project without needing to duplicate all files, assets, and game scenes. Instead it establishes pointers to the original file location with read only access. Because of this, it is safe and easy to create, load and destroy as many clone environments as required.

6.4 Performance & System Testing

Unity's integrated profiler was used to analyse the performance of the application. Since this is a time-consuming process, it is recommended to complete this at the end of a significant feature implementation [60]. Profiling was done comparatively in that sections of the game were recorded, saved, and analytically compared to its previous performance. It is wise to profile after an extended amount of time so that the application has reached a sustained state [61].

The Unity profiler is an instrumentation-based profiler [60]. At each stage, a 'snippet' of 300 frames of is captured by inserting a small marker in each function call – and calculating the execution time of this function. Although this creates a tiny bit of overhead, the overall profile is unchanged. Figure 52 illustrates the overall frame rate captured which sustained a mean average of 7.03ms (142 frames per second).

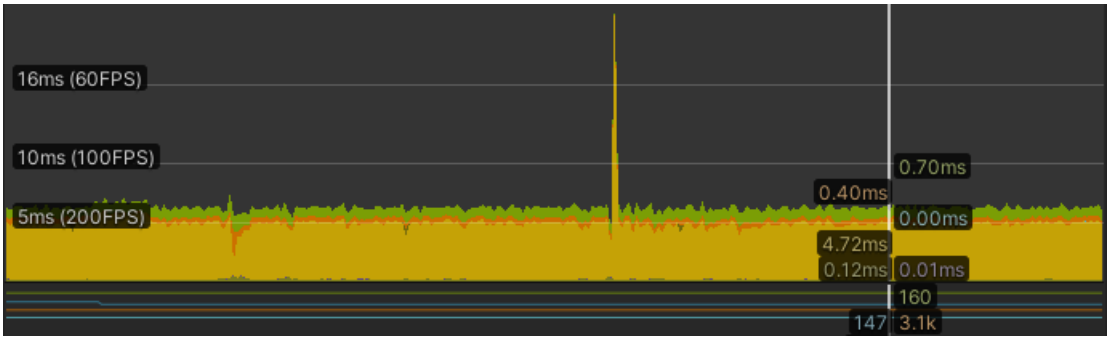


Figure 52: Average FPS of a 300-frame sample

6.5 Achieving a Target Frame Rate

In each frame, the core update loop is processed by the CPU and will require a certain number of milliseconds to complete; this time is inversely proportional to the frame rate [61]. Therefore, to achieve a target frame rate, the core update loop must be less than, or equal to, a target budget which is given by:

$$Budget = 1000\ milliseconds(ms)\ /\ Target\ Frame\ Rate$$

To achieve 60 FPS, the target budget would be equal to: $1000\ ms\ /\ 60 = 16\ ms$. Therefore, the application's target budget is 16ms - such that each frame within the player's update loop is executed in 16ms or less consistently.

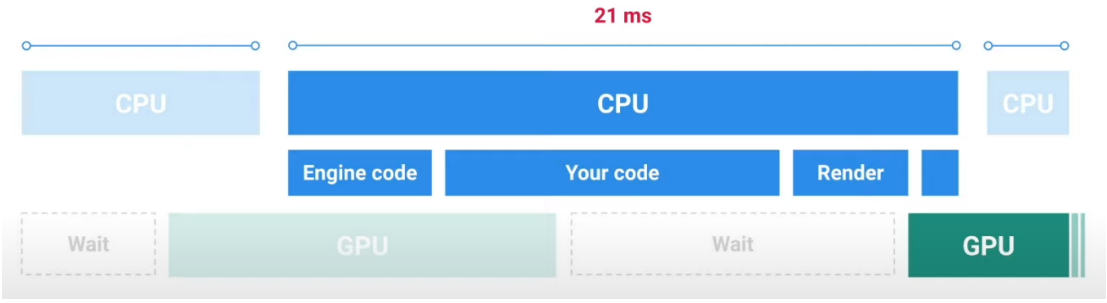


Figure 53: Structure of a Frame. Source - [61]

6.6 Evaluation and Optimisation

The first significant optimisation re-worked how tiles updated their status (Appendix 10.2.4). Since the game-scene had 135 tiles, there were 135 tile updates per frame which occupied 0.15ms (0.9%) of the target budget (Appendix 10.2.2). Because the application should scale for new scenarios, if this scenario was scaled to 1350 tiles, this would become 9% of the target budget. To amend this, tiles were re-worked such that other classes invoke a tile status change only when necessary. One down-side of this approach is that it couples the tile component more tightly to other classes.

The second significant optimisation fixed unnecessary memory losses as well as visual bug that caused units to 'shake' within the WebGL version (Appendix 10.2.5). From profiling it was observed that the garbage collector was invoked 544 times to re-allocate 20 kilobytes (kb) of memory per frame (see Figures 54 and 55). This resulted from BFS calculations initiating 493 ray-casts each frame. This is significant because ray-casting is a relatively expensive operation, and the number of ray-casts is proportional to the number of tiles. At 135 tiles, this was occupying 0.89ms (5.3%) of the target budget. To amend this, tile data was cached upon initialisation and BFS calculations were reduced to only be invoked once upon selecting a unit.

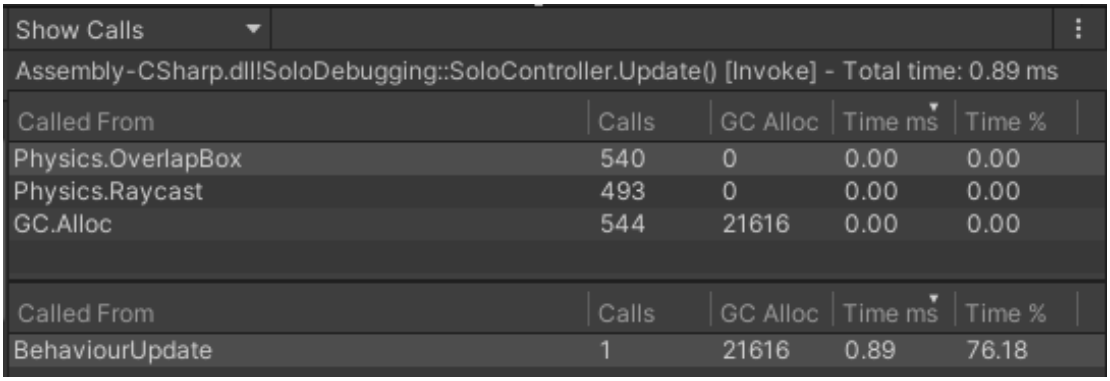


Figure 54: 544 Garbage Collection Allocations per Frame (21kb)

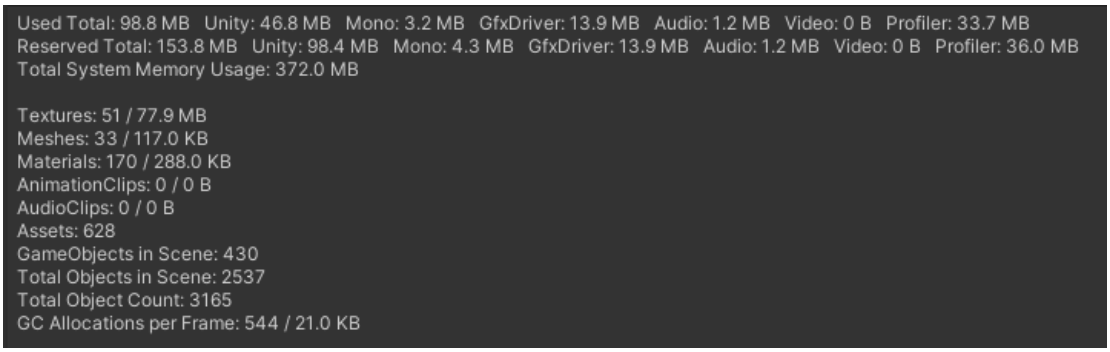


Figure 55: Memory Usage (per 300 frame capture)

After thoroughly optimising the game, the mean frame budget was lowered to 6.97 (1.69% less per frame), with no lag spikes exceeding the target budget of 16ms (Appendix 10.2.5).

6.7 Evaluation of Networking Strategy

Photon Cloud and PUN2 proved to be incredibly optimised for creating a cross-platform multiplayer application. Regarding performance and frame rate, there was negligible impact in single-player mode vs multiplayer. Figure 56 illustrates the performance impact of using the PUN2 API per 300 frame sample, which had a mean budget of 0.02ms per frame (0.13% of our target). See Appendix 10.2.6 for more details. One caveat of Photon Cloud is that matchmaking can only occur within server clusters located in the same region.

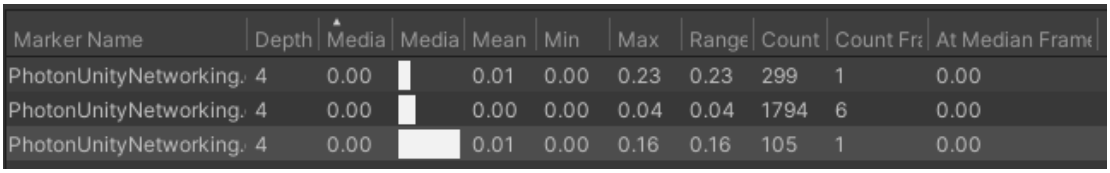


Figure 56: Photon Networking Impact Per 300 Frame Sample

6.8 Evaluation of Application Frame Rate

Table 5 illustrates the average, minimum and maximum frame rate achieved on different platforms. Each test was recorded after 2 minutes to achieve a sustained state.

Platform	Avg FPS	Min FPS	Max FPS
Windows (PC)	144	97	160
Microsoft Edge	136	96	135
Google Chrome	133	109	141
Firefox	71	56	73
Safari (iPad 2017)	42	22	45
Google Chrome*	12	10	24

Table 5: Performance on Different Platforms

From Table 5, it is evident that Google Chrome and Microsoft Edge heavily support WebGL as the performance on these browsers matched that of the Windows application (used as a benchmark). Note these frame rates are only achievable from enabling hardware acceleration which WebGL requires to access the system's dedicated graphics processing unit [41].

The final row in Table 5 demonstrates the frame rate in Google Chrome without hardware acceleration which exemplifies its importance. Unfortunately, Internet Explorer did not support WebGL.

It must be noted that these frame rates are dependent on the hardware available. However, even on weaker devices, 30fps was still achieved which is the industry standard.

6.9 Evaluation of Project Completion

Table 6 presents the completion of all user stories established from the project requirements.

This project was successful as all 'musts' were completed with exception of task 28. Task 28 refers to how effective the application is at teaching the OWASP security risks which can only be answered from validation testing. User feedback was not prioritised in this project as it was out of scope of the MDA framework. Regarding this however, the Design-Dynamics-Experience framework [62] extends upon this framework by evaluating the user experience from the existing design and dynamics. Unfortunately, this approach requires more time as it adds more components into the overall development process.

All 'shoulds' were fully implemented, bar task 18 which was partially implemented through a tips & hints system. A full tutorial implementation was prioritised last in the development cycle as it was identified in the literature review to not be a core necessity.

Regarding the incomplete 'coulds', tasks 29 and 30 are befitting for future work. Task 29 stems from the application-requirement of being extensible to creating new scenarios, units, abilities and adding more mechanics in general. In task 30, it was realised that developing an automated feedback system would have a similar complexity to that used in Chess. However, Chess has a rigorous history and ruleset which this project does not. Therefore, such a feature would not be possible given the timespan of this project.

ID	Functionality or Feature	MoSCoW	Completion
1	View a unit's attributes	Must	Yes
2	View an ability's attributes	Must	Yes
3	View a colour-blind friendly UI	Should	Yes
4	View a unit's movement range	Must	Yes
5	View a unit's ability attack range	Must	Yes
6	Move a unit	Must	Yes
7	Attack a target	Must	Yes
8	Disable a target unit	Must	Yes
9	Defend an ally unit	Must	Yes
10	Click on a move description	Must	Yes
11	Click on a unit	Must	Yes
12	View a history log	Should	Yes
13	View the units remaining	Should	Yes
14	Pan the camera and/or zoom in	Could	No
15	Unique animations for each ability	Could	No
16	Varied unity abilities	Could	Yes
17	Cost requirement for each ability	Should	Yes
18	An intermediate tutorial	Should	Partial
19	Hints and tips bar	Should	Yes
20	Multiplayer functionality	Must	Yes
21	Single-player functionality	Won't	Yes
22	Resign and quit	Must	Yes
23	Win or lose	Must	Yes
24	Play on Desktop	Must	Yes
25	Play on Web Browser	Should	Yes
26	Play on Tablet	Should	Yes
27	Play cross-platform sessions	Could	Yes
28	Learn about the OWASP Top Ten	Must	Maybe
29	Purchase and upgrade new units	Could	No
30	Receive feedback after a game	Could	No
31	Hear gameplay music	Should	Yes
32	Able to create new scenarios	Could	Yes

Table 6: Completion Analysis of User Stories

7 Project Management

Table 7 describes the tools, techniques and software used to complete this project.

Tools	Description
GitLab	Project management (Boards, Issues and Milestones)
GitHub Desktop	Local software for handling version control
Trello	Lightweight tool for managing daily and weekly tasks
Workona	Chrome Extension for organising tabs into a succinct workplace
Menderley	Reference management software for literature research
Microsoft Teams	Communication software for weekly remote meetings
Google Drive	Cloud storage for research documents and recording minutes
Lucid Chart	Online software for UML diagrams
Unity	Game engine for developing for PC and the Web (WebGL)
C#	Primary programming language for Unity
JetBrains Rider	Integrated development environment for C# & Unity
Photon Cloud	Cloud service for hosting multiplayer servers online
PUN2	Multiplayer networking API
ParralelSync	Open source software to clone the Unity development environment
Itch.IO	Secure domain to host WebGL sessions online

Table 7: Project Management Tools & Techniques

7.1 Reflection of Project Management

This project was completed with the Agile methodology in mind. As such, GitLab's Issues and Milestones were used to deliver weekly updates. This provided invaluable project feedback and assistance which ensured the development process did not deviate. Despite some personal difficulties in February to March, all major project requirements were completed in accordance to their MoSCoW priority.

7.2 Gantt Chart - Phase One

Table 8 demonstrates the planning and research completed during phase one.

Week Beginning	1	2	3	4	5	6	7	8	9	10
Date beginning	Oct5	Oct12	Oct19	Oct26	Nov2	Nov9	Nov16	Nov23	Nov30	Dec7
Planning										
Project Brief										
Literature Review										
Submit Project Brief		x								
Project Planning										
Identify Hypothesis & Research Question										
Research										
Review Gamification Mechanics										
Research Cyber Security Games										
Review Related Work										
Identify Project Requirements										
Brainstorm Game										
Write Report										
Submit Report										x

Table 8: Gantt Chart for Phase One

7.3 Gantt Chart - Phase Two

Tables 9 & 10 present the planned schedule and the actual outcome during phase 2. The 'implementation' tasks were adapted from the user stories (requirements) and scheduled in accordance to their MoSCoW priority, as well as their implementation priority. For example, units are required before unit abilities can be implemented.

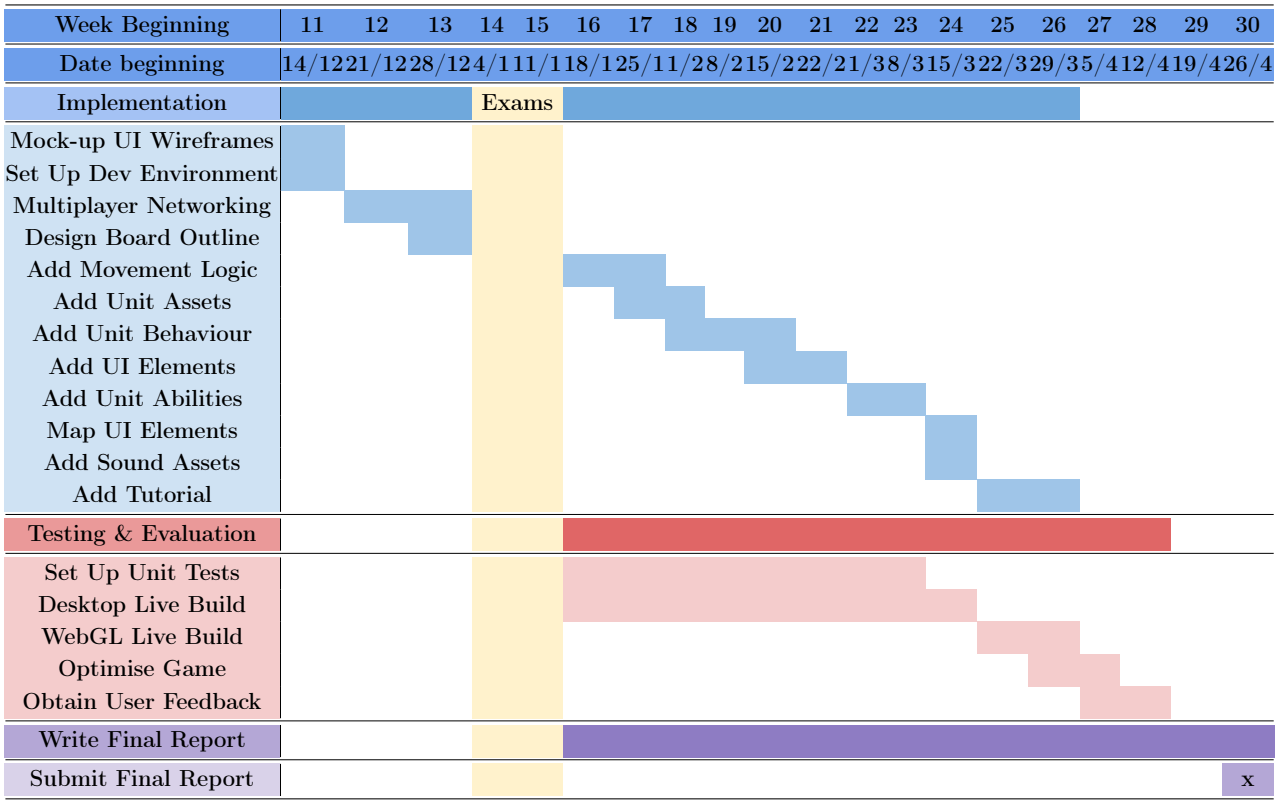


Table 9: Planned Gantt Chart for Phase Two

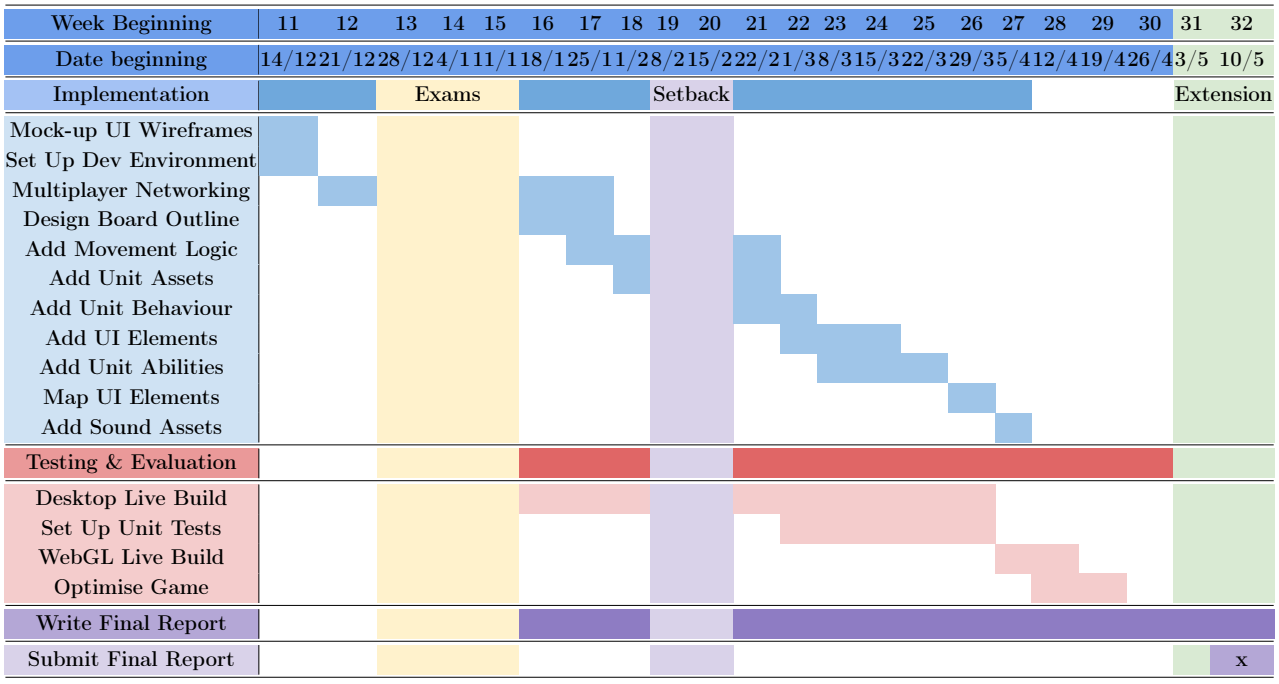


Table 10: Retrospective Gantt Chart for Phase Two

7.4 Risk Assessment

The following risk assessment ensured appropriate actions were taken for the successful completion of this project. The risk exposure (R.E) is given by the product of the probability and severity of each risk.

Risk	Prob (1-5)	Sev (1-10)	R.E	Mitigation
Project deadlines not met	3	10	30	Establish soft deadlines and meet with supervisor weekly for continuous feedback and self-evaluation
Online multiplayer failure	3	9	27	Develop LAN multiplayer with Mirror, or local multiplayer co-op
Application not effective at teaching	3	7	21	Identify the appropriate target demographic and cyber security content to teach
Over-estimating scope of implementation	3	7	21	Plan project into smaller iterations and prioritise according to MoSCoW
Application does not meet original problem & hypothesis	2	9	18	Refer to the initial problem statement and hypothesis
Application does not meet user requirements	2	8	16	Prioritise 'musts', 'shoulds' and then 'coulds'
Sickness, flu, or mental health difficulties	2	8	16	Exercise daily, eat healthy, and reach out for support if needed
Impact of Covid19	5	3	15	Use online multiplayer for remote gameplay, alongside Microsoft Teams & GitLab for communication
Gamified mechanics are poorly implemented	2	7	14	Identify key mechanics through literature and game review, and use the MDA framework to design the application
Stolen work or data	1	10	10	Use a secure password manager, 2FA, and back up important files
Loss of work or data	1	10	10	Use Git version control and One Drive/Google Drive for cloud storage
Under-estimating scope of implementation	1	5	5	Refine the game after evaluation, and implement all 'coulds' if time permits

Figure 57: Risk Assessment

8 Conclusion and Future Work

This project set out with the following research question and hypothesis:

Research Question

Can teaching cyber security through a gamified medium improve cyber security awareness?

Hypothesis

The MDA framework will satisfy the educational requirements for creating a serious game

Regarding the research question, a thorough review of pre-existing cyber security games, related work and literature review was completed to answer this. It can be concluded that gamified strategies can improve cyber security awareness, with the OWASP Cornucopia exemplifying this [26].

Regarding the hypothesis, combining the MDA framework with a variety of gamified mechanics, identified in Forde's report [13], creating a game with educational elements was completed in time.

However, because designing fun games is difficult, with educational games being even more challenging, a project like this requires a more flexible schedule such that each iteration of design can be periodically reviewed and tested from user feedback. As such, a framework more tailored for educational game design, such as the Design, Dynamics, Experience framework [60], would be befitting for this project going forward.

Through consistent project planning and utilisation of Unity and Photon Cloud, all major tasks were completed. Furthermore, all project goals and learning objects outlined in the project brief were achieved (Appendix 10.6). The minimum viable product created can be played above 30FPS on all platforms tested and up to 140FPS on Google Chrome. Multiplayer matchmaking supports cross-platform play and can be accessed without server hardware. This project also supports the three most popular aspect ratios and can be fully played with touch-screen controls.

Although only one scenario was created to exemplify the OWASP Top Ten [6], this project can easily be used as a framework to build further scenarios without any programming requirement. This is possible because units, abilities, tile objects, and user interface components, are all modelled as 'Prefabs'. This means all parameters for unit attributes, abilities, and OWASP information, can be freely changed within the inspector. To summarise, only the aesthetics and physical design needs altering.

8.1 Future Work

Additional Scenarios

Additional scenarios and levels would provide more out of the box experiences and interactivity with the application.

Custom Level Editor

A custom level editor would allow players to create their own in-game scenarios, without Unity, for the purpose of training.

Upgradable Units & Abilities

An upgrade skill tree for unit abilities, or unit attributes, would add an extra depth of strategy to the pre-existing mechanics.

Evaluation from Feedback

To test if learning objectives are met, receiving feedback on the existing design and dynamics would be necessary to verify the application's potential.

Single-player Mode & Story

A single-player story with an objective would engage users who cannot play multiplayer.

Additional Gamification Mechanics

In game quizzes could be set between turns which test a player's knowledge. This would provide feedback and a competitive advantage (or disadvantage) upon answering.